Universidad
Carlos III de Madrid

Grado en Ingeniería Informática

Trabajo Fin de Grado

Estudio de técnicas de reducción de dimensionalidad para problemas supervisados

Autor: Ignacio Segovia Lasanta

Tutores: Ricardo Aler Mur

José María Valls Ferrán

Junio 2015

Índice de contenido

Índice de contenido

Índice de Tablas

Índice de figuras

Índice de ecuaciones

Agradecimientos

Agradecimientos

Quiero hacer una mención especial a todas aquellas personas que han estado a mi lado durante todo este tiempo.

Los primeros y más importantes son mis padres a los que les estoy enormemente agradecido por el esfuerzo que han realizado y todas las ayudas facilitadas para que a día de hoy haya podido conseguir todos los objetivos que me he propuesto. Gracias.

La siguiente que necesita una mención especial es Marta, presente todos los días, todos los meses, todos los años… presente tanto en los malos momentos como en los buenos, siempre dispuesta a ayudar. Presente aunque haya una gran distancia entre nosotros. Gracias. Nunca podré agradecértelo tanto como te mereces.

A mis amigos, que siempre están ahí ayudando o molestando, pero haciéndome la vida un poco más alegre. A vosotros… no a vosotros no os agradezco nada, no lo merecéis.

A mis compañeros de carrera. Demasiadas horas juntos intentando sacar las cosas adelante, prácticas interminables, tardes de estudio interminables hacen que seamos una pequeña familia, sinceramente sin vosotros esto no hubiera sido posible. GRACIAS. Espero que esta relación dure mucho tiempo y no perdamos el contacto con el paso del tiempo. Sois geniales.

A Gumpus, gracias a ellos puedo hacer música, disfrutar de la música, y hacer que otros disfruten de ella (o tengan que mentir cuando son preguntados). Esperemos que el proyecto siga adelante y podamos seguir dedicándole el tiempo que se merece. Gracias.

Por último y no por ello menos importante quiero agradecer a mis tutores del trabajo de fin de grado: José María Valls y Ricardo Aler. Muchas gracias por confiar en mí y gracias por todo lo que me habéis enseñado y ayudado durante todo este tiempo.

Capítulo 1: Introducción y objetivos

1. Introducción y objetivos.

1.1 Introducción.

En lo que se refiere a la ciencia de la computación uno de los principales desafíos que siempre se ha tenido es conseguir construir máquinas que tengan la capacidad de aprender, que sean capaces de mejorar de manera automática basándose en su experiencia, ya que de conseguirse, se abriría un mundo de desarrollo y posibles nuevas aplicaciones. Por otro lado, también ayudaría a comprender los límites del aprendizaje humano.

Los seres humanos somos capaces de aprender y razonar gracias a la red de neuronas que tenemos en el cerebro. Las redes de neuronas artificiales están inspiradas en la forma en la que el sistema nervioso de los animales funciona. Es un paradigma de aprendizaje y procesamiento automático mediante el cual una serie de entradas se transforman y generan una o varias salidas. Dicha técnica se utiliza con mucha frecuencia en los diferentes campos de la ingeniería y la ciencia para resolver problemas complejos en los que las técnicas más tradicionales como pueden ser la regresión lineal o polinómica no funcionan todo lo bien que se espera de ellas.

Las redes de neuronas crean un modelo no explícito que relaciona el conjunto de entradas para poder generar unos valores de salida. Para ello es necesario un conjunto de observaciones de las variables de entrada, con las cuales se crean unos patrones de entrenamiento que permiten que la red sea capaz de aprender y capaz de predecir una salida cuando se introducen nuevas observaciones.

Es importante conocer cómo funciona el aprendizaje de las redes de neuronas. El aprendizaje de las redes de neuronas construye modelos formados por varias capas: capa de entrada, capas ocultas y capa de salida. Las arquitecturas más utilizadas tienen una única capa oculta, que se puede entender como una transformación de los atributos de entrada (capa de entrada) en una representación intermedia (capa oculta) que facilita la clasificación del dato en la capa de salida.

Por ejemplo, en una red en la que la capa de entrada está formada por 10 neuronas, la capa oculta tiene 2 y la capa de salida está formada únicamente por una neurona, se puede entender como una transformación de los 10 atributos de entrada en 2 atributos intermedios que utiliza la red para generar la salida. Normalmente, el objetivo del aprendizaje de las redes de neuronas es la salida que producen. Sin embargo, para este trabajo de fin de grado se quiere aprovechar la capa oculta de la red de neuronas como un mecanismo mediante el cual se realice una transformación de atributos y una reducción de dimensionalidad. Si nos basamos en el ejemplo anterior, se transformaría un problema de 10 atributos en un nuevo problema de 2 atributos.

Aunque existen técnicas de transformación y reducción de dimensionalidad como por ejemplo PCA (Principal Component Analysis), ésta tiene el problema de ser técnicas no supervisadas, en las que al hacer la transformación tienen en cuenta los atributos de entrada, pero no la salida. En este trabajo nos interesan los problemas supervisados, en los que se quiere predecir una salida a partir de una entrada. Por tanto las transformaciones que produce PCA, puede que no sean las más apropiadas (a pesar de ser una de las técnicas de reducción más utilizadas).

La capa oculta de las redes de neuronas, se crea como resultado de un proceso de aprendizaje supervisado, es decir, un aprendizaje que tiene en cuenta la salida. Por lo tanto es posible que en la capa oculta se generen transformaciones y reducciones de dimensionalidad que sean capaces de mejorar a otras generadas por técnicas no supervisadas como PCA.

Esa comparación es la que se quiere llevar a cabo en este trabajo de fin de grado. Para comprobar la eficacia de ambas técnicas, se las utilizará en combinación con otra técnica de aprendizaje automático que es sensible a la presencia de atributos irrelevantes: KNN (K-nearest neighbours) o K-vecinos.

1.2 Objetivos.

El objetivo de este trabajo es utilizar diferentes técnicas de transformación y reducción de dimensionalidad sobre varios conjuntos de datos para poder compararlas y observar si pueden recoger buenos resultados clasificando nuevas instancias utilizando un menor número de atributos de entrada.

Por tanto, este proyecto es un trabajo de investigación en el que no se conoce si los resultados que se van a obtener serán favorables o no. Sin embargo, si los resultados son positivos, se abre la puerta para investigar más a fondo o comenzar a utilizar estas técnicas de reducción a la hora de necesitar resolver problemas con un número elevado de atributos.

Es necesario recalcar que en la época actual el Big Data está en un momento ascendente, y a la hora de resolver problemas, poder reducir el tamaño de los datos sin perder información es bastante importante.

Los objetivos de este trabajo son:

- Construir un sistema que permita entrenar una red de neuronas y extraer la transformación realizada por su capa oculta.
- Comparar la transformación realizada por la capa oculta de la red con la transformación PCA. Utilizando para ello un conjunto de dominios y el algoritmo de clasificación KNN.

1.3 Estructura del trabajo.

El trabajo está dividido en 9 capítulos, en el primer capítulo se muestra la introducción al trabajo así como los objetivos del mismo.

El segundo capítulo está formado por el estado del arte, tiene como cometido poner en situación al lector e informarle de todos los conocimientos que son necesarios para poder entender el trabajo que se va a realizar.

En el tercer capítulo se muestra el diseño del sistema y se explica cómo se ha llevado a cabo las diferentes funciones que se utilizan para el estudio realizado.

En el capítulo 4 se explican las características de los dominios con los que se va a trabajar, la metodología, y las pruebas. También se muestran los resultados obtenidos de dichas pruebas y un análisis de éstos resultados.

En el capítulo 5 se expone la planificación y el presupuesto necesario para la realización de este trabajo de fin de grado.

El capítulo 6 está encarado a mostrar el marco regulador que rige este trabajo.

En el capítulo 7 se encuentran las conclusiones del estudio realizado, las conclusiones personales del autor, así como los futuros trabajos que se pueden realizar en base a los resultados obtenidos.

En el capítulo 8 se muestra la bibliografía consultada para poder realizar el trabajo.

Por último el capítulo número 9 consta de un resumen en inglés formado por un apartado de introducción y objetivos, un apartado de resultados y un apartado de conclusiones.

Capítulo 2: Estado del arte

2. Estado del arte.

2.1 Fundamento biológico de las redes de neuronas.

Las redes de neuronas artificiales surgen al observar el funcionamiento del sistema neuronal del cuerpo humano. Dicho sistema se puede dividir en 3 partes fundamentales:

- Órganos receptores: Su función es recoger la información exterior que rodea al cuerpo humano.
- Sistema nervioso: Recoge la información facilitada por los órganos receptores, la analiza, transmite, en ocasiones almacena y envía la información elaborada.
- Órganos efectores: Son los encargados de recibir la información del sistema nervioso y convertirla en una acción determinada.

El componente más importante del sistema nervioso es la neurona, que al unirse con otras crea una red de neuronas. Una neurona está compuesta por cuerpo, axón y las dendritas.

El núcleo, también denominado cuerpo, es la parte central de la neurona. El axón es la ramificación de salida de la neurona, mientras que las dendritas son el conjunto de ramificaciones que la neurona tiene como entradas.

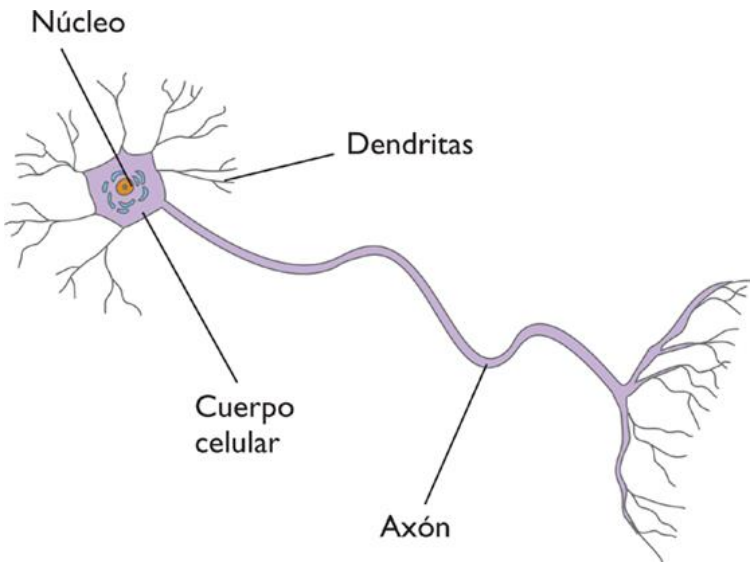

Figura 2.1 Componentes de una neurona biológica

El funcionamiento de una red de neuronas es el siguiente: las señales de entrada a la neurona llegan a través de la sinapsis, que es el proceso que pone en contacto las neuronas; la sinapsis toma información de las células cercanas que están conectadas con la neurona; dicha información llega por medio de las dendritas al núcleo de la neurona, que se encarga de procesar la información y de generar la respuesta que propagará a través del axón.

La sinapsis es un proceso que transforma una señal eléctrica en otra química. Somos capaces de aprender gracias a un sistema neuronal que es capaz de recoger un conjunto de información, procesarla y transmitirla. Y por parte de la siguiente neurona conectada potenciarla o inhibirla.

2.2 Historia de las redes de neuronas artificiales.

El principio del siglo XIX es el momento en que aparecen las primeras investigaciones acerca de las redes de neuronas artificiales, motivadas por los trabajos que Freud realizó.

Sin embargo, aunque fue Russell (1913) el primero en crear un dispositivo hidráulico capaz de imitar a una red de neuronas, no fue hasta los años cuarenta del siglo XX, cuando las redes de neuronas artificiales cobraron algo de fuerza. A partir de ahí, y gracias a la evolución del hardware, los científicos pudieron ahondar en la técnica e ir evolucionando en su estudio.

En los años 40 y 50, Warren McCulloch y Walter Pitts trabajaron en el que se convertiría en el primer modelo matemático de una red de neuronas, en el cual ya se introdujo la idea del paso por umbral que se utiliza en multitud de modelos actuales. Este modelo se conocería como el modelo McCulloch-Pitts.

A partir de este hito comenzaron a aparecer otros científicos que trabajando sobre estas ideas conseguían ir haciendo progresos en un área totalmente nueva.

En 1951 Marvin Minsky, basándose en el trabajo realizado por McCulloch y Pitts, se unió con Edmons y diseñaron la primera máquina en la que las conexiones se iban ajustando dependiendo de los resultados obtenidos al realizar determinadas tareas.

En 1956 Albert Uttley desarrolló nuevos paradigmas que eran capaces de ajustar los parámetros de entrada utilizando la medida de entropía de Shannon. Gracias a ello, se pudo comenzar a simular fenómenos atmosféricos y el reconocimiento adaptativo de patrones.

En 1957, Frank Rosenblatt generalizó el modelo de McCulloch y Pitts añadiéndole aprendizaje, y lo denominó Perceptrón (modelo utilizado en el estudio de este trabajo). El problema con el que se encontró Rosenblatt fue que no encontró un modelo matemático capaz de trabajar con su procedimiento de aprendizaje (basado en el ajuste de pesos de las conexiones entre capas) que fuese capaz de trabajar más allá de dos capas.

Al cabo de dos años, en 1959, Bernard Widrow consiguió crear una red de neuronas artificiales capaz de ajustar los pesos entre los distintos niveles basándose en el error obtenido (Adaline). Este modelo también estaba limitado a dos capas.

En la década de los 60 muchos investigadores desarrollaron nuevos métodos. Steinbuch, con las redes de Steinbuch, consiguió que se pudiera aplicar esta técnica al reconocimiento de escritura a mano distorsionado, al diagnóstico de fallos de maquinaria y al control de procesos de producción. Grossberg, que fue de los más influyentes de la época, realizó numerosos estudios sobre los procesos de procesamiento humano de la información, que al unirlos con análisis matemáticos permitieron el nacimiento de nuevos paradigmas de redes de neuronas, mediante los cuales se puede acceder a la información mientras que ésta va siendo procesada.

A finales de los años 60 y comienzos de los 70 se produjeron muchas investigaciones en torno a las redes de neuronas artificiales. Cabe destacar a Shun-Ichi Amari (gracias a su estudio se dio solución al problema de la asignación de créditos, irresoluble hasta entonces),

Longuet y Higgins (fueron los creadores del primer sistema de ecuaciones codificadas que era capaz de almacenar y recuperar una secuencia formada por señales), Kunihiko Fukishima (Cognitron y neocognitron) y Harry Klopf (sus estudios relacionando la biología cerebral con la psicología de la mente revelaron que la neurona es un componente que se mueve en busca de metas).

Por último, en las décadas de los 70 y 80, Teuvo Kohonen comenzó a crear paradigmas de conexiones aleatorias, y junto a Ruohonen comenzaron a extender el modelo de la memoria lineal asociativa, creando OLAM (asociador óptimo de memoria lineal). Posteriormente desarrolló el famoso LVQ (Learning Vector Quantization). Leon Cooper y Charles Elbaum fueron los que primero comercializaron una patente de redes de neuronas artificiales bajo el nombre de Nestor Associates. Su éxito comercial más importante fue RCE, con su modelo propio de aprendizaje SNL.

Otros investigadores y científicos que trabajaron en la materia, ya sea de manera matemática o psicológica, son Terence Sejnowski, McClelland y Rumelhart (con su paradigma de reconocimiento de voz), Jerome Feldman y Dana Ballard, Robert Hecht-Nielsen (por crear el primer computador neuronal), John Hopfield o Bart Kosko.

2.3 Red de neuronas artificiales.

2.3.1 Componentes de una neurona artificial.

La idea principal de las redes de neuronas artificiales es tratar de imitar el funcionamiento de una red de neuronas biológica.

Para ello se necesitan unas unidades centrales que se denominan neurona, que son las encargadas de recibir, transformar mediante una función de activación y enviar la información transformada para generar una salida.

Al igual que las neuronas biológicas, las neuronas artificiales están conectadas unas a otras. Cada una de las conexiones tiene un peso, cuyo objetivo es ponderar cada conexión o entrada a la neurona. La entrada a una neurona es la suma de las salidas de la neurona anterior multiplicada cada una de ellas por su respectivo peso.

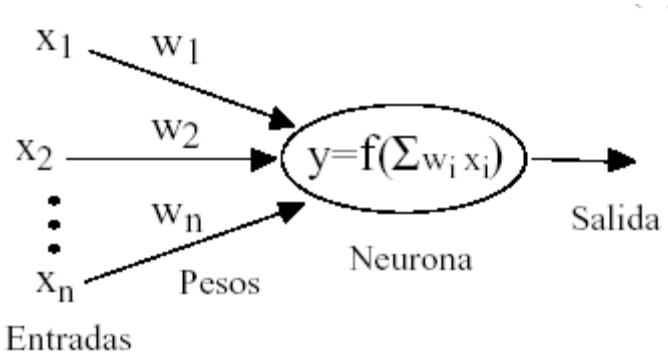

Figura 2.2 Esquema de una neurona artificial

En la figura 2.2 se puede observar que la neurona recibe un conjunto de entradas $X_1,X_2,\ldots,X_n$. Cada una de ellas es multiplicada por el peso establecido para esa conexión $W_1,W_2,\ldots,W_n$. A continuación se produce el sumatorio de todas las multiplicaciones y se les aplica una función de activación f(). Es importante destacar que los pesos comienzan con unos valores aleatorios y que son ajustados en la fase de aprendizaje de la neurona.

Las neuronas están conectadas creando una arquitectura formada por tres partes, entradas, salidas y capas ocultas, que dependiendo de la arquitectura que se utilice puede ser de una o varias capas a su vez. Dependiendo del problema que se quiera tratar se crean distintos tipos de arquitectura, y mediante observación se comprueba cuál es la que se adecúa mejor. No hay un tipo de arquitectura que sea óptimo o general para todos los problemas. Por eso la decisión de crear una arquitectura concreta no garantiza que se vayan a obtener buenos resultados.

2.3.2 Funcionamiento de una red de neuronas.

La red comienza recogiendo los valores de entrada X_j y multiplicándolos por los respectivos pesos $W_{i,j}$ (donde los subíndices 'i' y 'j' indican la capa oculta a la que corresponde y el identificador del dato respectivamente). El resultado de dichas multiplicaciones será propagado a la siguiente capa, y por tanto la salida de cada neurona vendrá dada por la siguiente ecuación:

$$u_i(w, x) = \sum_{j=1}^{n} w_{ij} x_j$$

Ecuación 2.1 Salida neurona oculta

La salida que se observa en la ecuación 2.1 viene determinada por una función que se denomina función de activación, que se explicará en el apartado 2.3.3.

La salida de una neurona oculta se propagará a la siguiente capa de neuronas ocultas para seguir con su procesamiento. En caso de que esta segunda capa no exista, la salida de la neurona será el resultado final.

Para que el funcionamiento de la red quede de una manera más aclarada se presenta la figura 2.3 en la que está representada una arquitectura de red neuronal con una capa oculta.

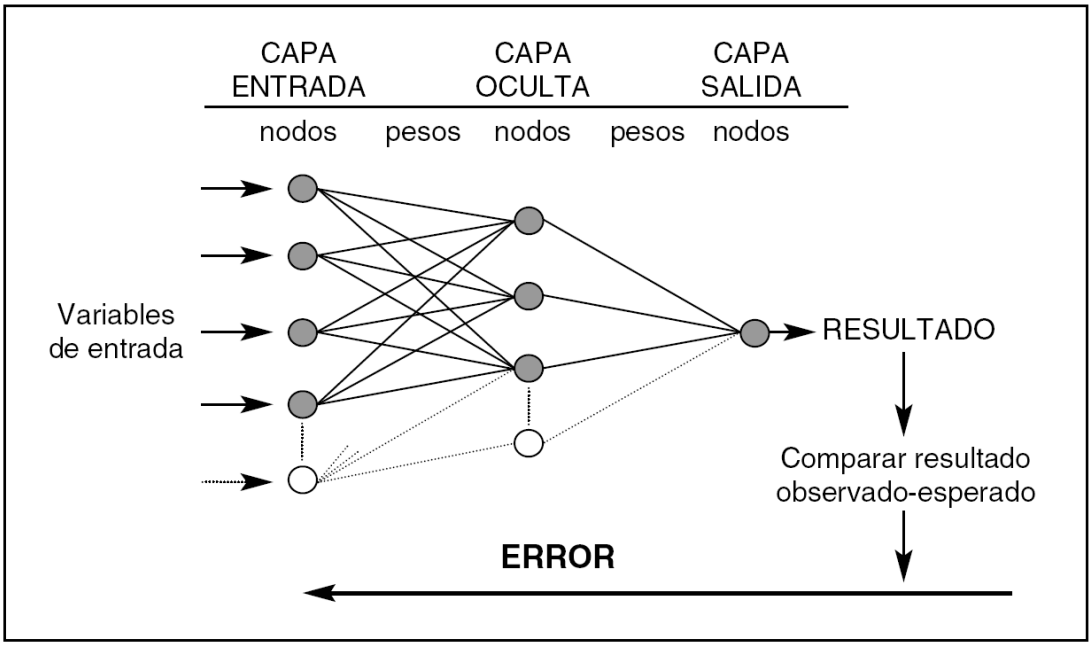

Figura 2.3 Funcionamiento de una red de neuronas

Como se observa, las variables de entrada se multiplican por los pesos y llegan a la capa intermedia que propaga los respectivos resultados. Estos resultados se multiplican por los

nuevos pesos y llegan la capa de salida, que tras procesar los datos recibidos facilita la salida o resultado.

2.3.3 Función de activación.

El principal objetivo de la función de activación de una neurona es ser la encargada de tomar la información recibida en las entradas y relacionarla con el siguiente estado de activación que tenga esa neurona.

Aunque no suele influir en la capacidad de la red para resolver un problema, hay que tomar la decisión en el diseño de la red de cuál seleccionar. Existen dos funciones de activación importantes: función sigmoidal o tangente hiperbólica. Ambas son crecientes con dos niveles de saturación, y su mayor diferencia es el rango en la que trabajan de [-1,1] o de [0,1].

Función sigmoidal

$$f_1(x) = \frac{1}{1 + e^{-x}}$$

Ecuación 2.2 Función sigmoidal

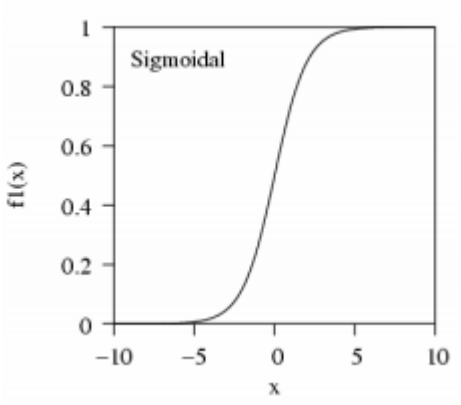

Figura 2.4 Función sigmoidal

Función Tangente Hiperbólica

$$f_2(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Ecuación 2.3 Función Tangente Hiperbólica

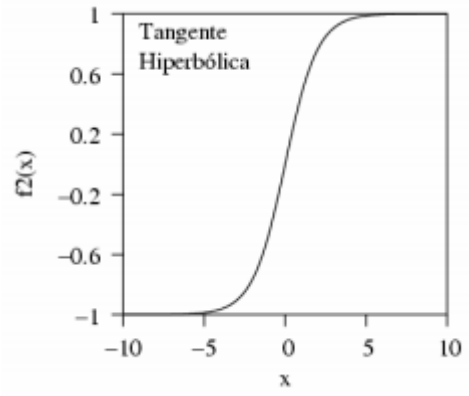

Figura 2.5 Función Tangente Hiperbólica 1

2.3.4 Creación y desarrollo de una red de neuronas.

El momento de crear y desarrollar una red de neuronas se puede dividir en los siguientes pasos:

- Diseño de la arquitectura.
- Entrenamiento.
- Prueba o test.

2.3.4.1 Diseño de una arquitectura de red de neuronas.

En esta fase se han de tomar las decisiones correspondientes a las características que la red ha de tener a la hora de trabajar. Se ha de determinar el número de neuronas que se van a utilizar y el número de capas ocultas en las que dichas neuronas van a ser distribuidas.

Los parámetros variarán dependiendo del problema que se quiera solucionar. También hay que decidir la función de activación que se desea usar.

La arquitectura de la red que se crea tiene que estar adecuada al problema. En caso contrario la red pierde capacidad de aprendizaje y los resultados que conseguirá no serán los mejores posibles. En ocasiones se ha de recurrir a la técnica de ensayo y error para conseguir los mejores parámetros.

2.3.4.2 Entrenamiento.

Con una arquitectura ya diseñada, el siguiente paso es el entrenamiento, que es el encargado de conseguir que la red consiga recoger información suficiente de los datos de entrenamiento para que sea capaz de generalizar, es decir, de comportarse adecuadamente ante los nuevos datos que se facilitarán en la fase de test.

Para alcanzar este objetivo, los pesos en esta fase van cambiando para ajustarlos y conseguir que la red cumpla los objetivos de generalización y lograr los mejores resultados.

2.3.4.3 Prueba o test.

Una vez creada y entrenada la red, se toman nuevos datos de entrada. Es importante que estos nuevos datos no se hayan utilizado en el entrenamiento. Deben ser datos nuevos, para poder comprobar de la manera más fiable posible, si la red ha sido capaz de generalizar y capaz de conseguir buenos resultados.

Los nuevos datos se facilitan a la red y ésta los ejecuta y calcula la salida. A continuación se comparan las salidas de la red con las salidas esperadas y se comprueba la capacidad de generalización de la red.

2.4 Herramientas y métodos utilizados.

2.4.1 Tipos de aprendizaje.

Existen dos tipos de aprendizaje para una red de neuronas: aprendizaje supervisado y aprendizaje no supervisado.

2.4.1.1 Aprendizaje supervisado.

En el aprendizaje supervisado, disponemos siempre de las salidas que se esperan de la red y por tanto se podrán comparar con las salidas que la red obtiene y calcular el error producido. Dicho error nos guiará en el ajuste y nos dará una medida para la modificación de los pesos. El método de trabajo es el siguiente:

La red recibe las entradas, las transforma y consigue una salida (salida real) que se compara con la salida deseada (que es la salida que corresponde a los datos de entrada y que, si la red ha funcionado correctamente, coincidirá con la salida real). Al comparar la salida real con la salida deseada dependiendo del resultado se ajustan los pesos de red.

Es un proceso iterativo, se vuelven a ejecutar todos los datos de entrada hasta que la red detiene el entrenamiento. Es en ese momento cuando quedará completamente entrenada. Es decir, que ante un conjunto de datos de entrada, la red basándose en su aprendizaje, generará una salida que en el mejor de los casos será igual a la salida deseada.

2.4.1.2 Aprendizaje no supervisado.

Por el contrario, en el aprendizaje no supervisado, no se dispone de las salidas esperadas de los datos, por lo que no se puede tener una medida del error que la red comete.

Al desconocerse estos datos, la red intenta buscar regularidades y estructuras en los datos de entrada, como por ejemplo, agrupaciones o clusters.

Una de las ventajas del aprendizaje no supervisado es que no se influencia con información externa, pero como inconveniente tiene que se necesita un conjunto de datos grande, ya que la red se entrenará mejor cuantos más datos pueda analizar.

2.4.2 Redes de neuronas: Perceptrón multicapa.

El perceptrón multicapa se basa en el perceptrón simple, su existencia surge debido a las limitaciones que éste último presenta ante problemas de separabilidad no lineal. La solución se encontró uniendo varios perceptrones simples, dando como resultado al Perceptrón multicapa. Su primer objetivo una vez creado fue intentar que dicho sistema consiguiera resolver problemas no lineales. Esta idea sirvió de base para que Rumelhart, Hinton y Williams siguieran estudiando el problema, y sabiendo que en el perceptrón multicapa los pesos no se podían adaptar, (puesto que la regla que utiliza el perceptrón simple no es aplicable) consiguieron crear la regla delta generalizada que es la que se utiliza para funciones de activación no lineales y para las redes con varias capas.

El perceptrón multicapa es un aproximador, capaz de resolver problemas reales y que puede competir con cualquier aproximador universal. Todo debido a que es un método sencillo de utilizar y tiene una gran aplicabilidad. Además hay que añadir, que no se puede conocer de manera previa si un clasificador trabajará mejor que otro ante un nuevo problema, ya que los clasificadores funcionan de mejor o peor manera dependiendo del problema que tengan que solucionar.

El perceptrón multicapa también tiene inconvenientes, uno de ellos es que tiene un proceso de aprendizaje muy largo cuando se utilizan un gran número de variables, que hay que codificar los problemas reales utilizando valores numéricos (lo que produce que en algunas ocasiones sea dificultoso), y por último como el funcionamiento del perceptrón multicapa se basa en buscar relaciones entre las entradas y las salidas, con ciertos problemas su efectividad se puede ver mermada debido a las complicaciones con la que se encuentra el aprendizaje.

2.4.2.1 Estructura del perceptrón multicapa.

El perceptrón multicapa contiene tres partes fundamentales en su estructura, la capa de entrada, las capas ocultas (una o varias) y la capa de salida.

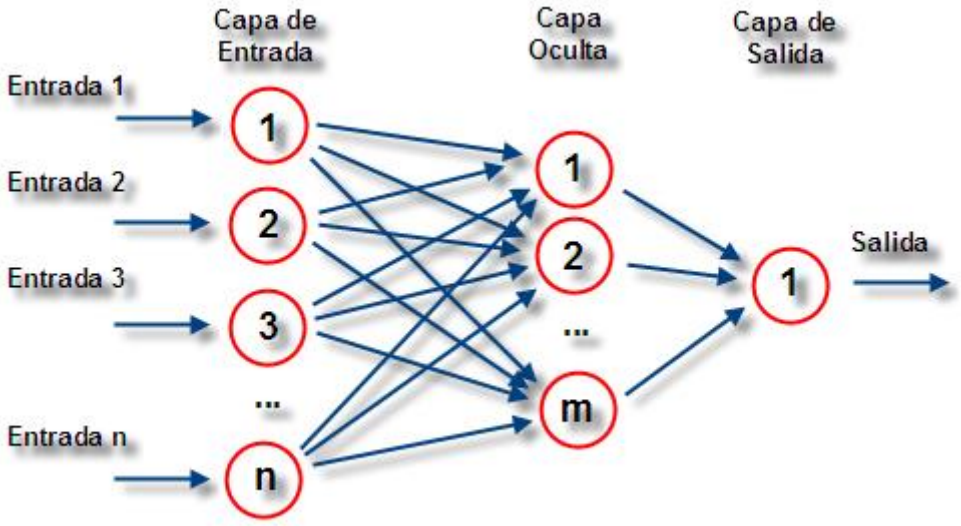

Figura 2.6 Estructura del perceptrón multicapa

La capa de entrada, aunque esté formada por neuronas, éstas no actúan como neuronas en sí, simplemente reciben los datos de entrada y los envían a la siguiente capa.

Si se observa la figura 2.6 se puede comprobar que las conexiones siempre avanzan hacia delante, de esta característica hereda el nombre este tipo de redes: redes hacia delante o "Feedforward".

Las conexiones entre neuronas de diferentes capas se denominan pesos y tienen un valor numérico real. Cada una de las neuronas tiene asociado un número adicional que se denomina umbral. Dicho umbral se considera como una conexión más a una neurona ficticia cuya activación es 1. Todas las neuronas de una capa están conectadas a las neuronas de la siguiente capa.

2.4.2.2 Función de activación del perceptrón multicapa y cómo se calcula.

Como se ha explicado en el apartado 2.3.3, la red de neuronas basa su funcionamiento en las funciones de activación, las cuales son elegidas por el usuario, pero normalmente se suelen utilizar una de las dos más comunes: la sigmoidal o la tangente hiperbólica. Ambas funciones están relacionadas entre sí y por lo tanto son semejantes en comportamiento.

A continuación se explica cómo se calcula la activación de las diferentes capas del perceptrón multicapa. Donde a_i^c representa las activaciones de la capa 'c' para cada elemento 'i'.

En la capa de entrada, las neuronas únicamente se encargan de enviar a la siguiente capa la información que les llega de los datos de entrada. Por lo que:

$$a_i^1 = x_i \; para \; i = 1, 2, \dots, n_1$$

Donde $X = (x_1, x_2, \dots, x_{n1})$ es el vector de entrada.

Ecuación 2.4 Activación de las neuronas de la capa de entrada.

Las neuronas que forman parte de las capas ocultas de la red son las encargadas de recibir la información, procesarla y aplicar la función de activación. Dicha función de activación se aplica al sumatorio del producto de las activaciones por sus respectivos pesos, y sumando el umbral, quedando la siguiente ecuación:

$$a_i^c = f\left(\sum_{j=1}^{n_{c-1}} w_{ji}^{c-1}\, a_j^{c-1} + u_i^c\right) para \; i = 1,2, \dots, n_c \; y \; c = 2,3, \dots, C-1$$

Donde a_j^{c-1} son las activaciones de la capa j-1.

Ecuación 2.5 Activación de las neuronas de las capas ocultas.

En cuanto a las activaciones de la capa de salida actúan de la misma manera que las neuronas de la capa oculta, es decir, aplican la función de activación al sumatorio del producto de las activaciones por sus respectivos pesos y seguidamente suma el umbral. Este resultado de las activaciones de esta capa, es el vector de salida de la red que contiene los resultados finales. Se formula con la siguiente ecuación:

$$y_i = a_i^C = f\left(\sum_{j=1}^{n_{C-1}} w_{ji}^{C-1}\, a_j^{C-1} + u_i^C\right) para\ i = 1,2,\dots,n_C$$

Donde $Y = (y_1, y_2, \dots, y_{nC})$ es el vector que contiene las salidas.

Ecuación 2.6 Cálculo de las activaciones de la capa de salida.

2.4.1.3 Aprendizaje del perceptrón multicapa.

El perceptrón multicapa es un algoritmo de aprendizaje supervisado, lo que quiere decir que para la fase de aprendizaje, modifica los parámetros en función de una salida deseada que ya conoce para que cuando se adapten dichos parámetros, la salida obtenida sea lo más cercana a la salida deseada. Por lo tanto lo que se realiza en el aprendizaje es un problema de minimización del error.

Para calcular el error medio para todos los patrones que entran en juego se puede utilizar la siguiente ecuación:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n)$$

Ecuación 2.7 Error medio de los patrones.

Donde N representa el número total de patrones que se utilizan en el aprendizaje y 'e(n)' representa el error que produce la red para cada patrón 'n'. 'e(n)' es el error cuadrático correspondiente a un patrón n, y se calcula sumando los cuadrados de las diferencias entre la

salida de la red y la salida deseada, para cada una de las n_C salidas (neuronas de la última capa). Se calcula de la siguiente manera:

$$e(n) = \frac{1}{2}\sum_{i=1}^{n_C}(s_i(n) - y_i(n))^2$$

Ecuación 2.8 Error producido para un patrón n.

Donde 's_i' representa la salida deseada ya conocida por la red y donde y_i representa la salida que calcula la red. 'n_C' es el número de neuronas de la última capa, es decir el número de salidas de la red.

A la hora de realizar una minimización no lineal las funciones de activación de tipo no lineal producen una respuesta no lineal con respecto a los pesos, lo que convierte el problema de minimización en un problema no lineal, en el que se ha de utilizar técnicas no lineales para su resolución.

Como hay que utilizar un método para la reducción del error, lo más estrictamente correcto sería tratar de minimizar el error total producido, sin embargo el método que se utiliza es el del gradiente estocástico. Consiste en ir minimizando de manera sucesiva el error que produce la red para cada patrón de entrenamiento. La manera de ir minimizando dicho error, es ir variando el valor de los pesos (w) de cada patrón de entrada siguiendo la ley de aprendizaje:

$$w(n) = w(n-1) - \alpha\frac{\partial e(n)}{\partial w}$$

Ecuación 2.9 Ley de aprendizaje.

Donde 'e(n)' representa el error cometido para el patrón n y α es la tasa de aprendizaje.

2.4.2 KNN: Vecinos más cercanos.

Otro método de clasificación utilizado en este estudio es el algoritmo conocido como KNN (k vecinos más cercanos). Es un algoritmo de los denominados perezoso, es decir, cuando realiza el entrenamiento, en lugar de construir un modelo como por ejemplo hacen las redes de neuronas o los árboles de decisión, guarda las instancias. La manera que tiene este algoritmo de clasificar consiste en: ante la aparición de una nueva instancia, ésta es clasificada como la clase mayoritaria de los vecinos más cercanos que entren en juego en el entrenamiento.

KNN es un algoritmo no paramétrico, no realiza suposiciones observando la distribución de los datos. Es un algoritmo local, lo que quiere decir que observa las clases a las que corresponden los vecinos más cercanos al dato analizado, y así decide asignar a la nueva instancia la clase mayoritaria entre dichos vecinos. Algunos algoritmos como pueden ser Fisher necesitan adaptarse cuando entran en juego dos o más clases, esto no le ocurre a KNN, lo que supone una ventaja.

Otra de las características de KNN, es que es un algoritmo sencillo, que se puede utilizar también en tareas de regresión calculando la media de los vecinos o una media ponderada por la distancia de cada vecino al dato.

Sin embargo, al igual que todos los algoritmos, no es perfecto y tiene ciertas desventajas a la hora de utilizarlo. Éstas se deben tener en cuenta.

Una de ellas es que en el momento que los datos que se utilizan poseen atributos irrelevantes, el algoritmo se muestra muy sensible y no produce buenos resultados. Ocurre lo mismo si los datos poseen ruido. A la hora de utilizar datos obtenidos del mundo real, se suelen presentar de una manera imperfecta en la cual existe información que sobra o que es incorrecta, es en ese momento cuando se dice que los datos tienen ruido. Es importante tenerlo en cuenta y está relacionado con el objetivo del proyecto, porque reducir la dimensionalidad en datos con ruido y va a poder mejorar la eficacia de KNN.

Es un algoritmo lento en el momento que entran en juego muchos datos de entrenamiento y además necesita que la función de distancia que se utilice sea la correcta. Generalmente la elegida es la distancia Euclídea, que viene definida por la siguiente ecuación:

$$d(P_1, P_2) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Ecuación 2.10 Ecuación de la distancia Euclídea.

Sin embargo todos estos problemas se pueden solucionar utilizando algunas técnicas. Al problema de la lentitud del algoritmo cuando entran en juego muchos datos, como solución se puede eliminar aquellos datos que sean superfluos. Eliminando aquellas instancias que tengan ruido se reduce el problema y como también es sensible a los atributos irrelevantes, eligiendo aquellos atributos entran en juego y apartando los datos del problema irrelevantes, quedaría solventado.

El punto crítico de KNN es el momento de elegir el número de vecinos que entran el juego, es decir el valor que va a tomar 'K'. Es un parte crítica a la hora de utilizar este método puesto que si se toma la decisión de utilizar únicamente el vecino más cercano, aquellas instancias que sean ruido tienen mucha influencia y no es algo favorable. Si se aumenta este valor el ruido va perdiendo influencia en el algoritmo, pero si se utiliza un número de vecinos muy alto, se pierde la idea de localidad y los vecinos más lejanos comienzan a tener influencia en un dato con el que puede que no tenga relación.

Otra característica que tiene el algoritmo KNN es que puede producir empates, por tanto si se van a utilizar dos clases, es necesario que el número de vecinos sea necesariamente impar, para así deshacer los empates y que el algoritmo pueda asignar una clase a la instancia utilizada.

Ejemplo gráfico.

A continuación se muestra un ejemplo gráfico del funcionamiento de KNN. En este caso es una clasificación entre seres humanos dependiendo de su altura y su peso, es decir que conociendo únicamente la estatura y peso de una persona se tiene que clasificar entre tres posibles opciones niño, adulto o mayor

Lo primero es recoger todos los datos de ejemplo clasificados correctamente. Se almacenan la altura, peso y se asigna la clase a la que representan esos datos.

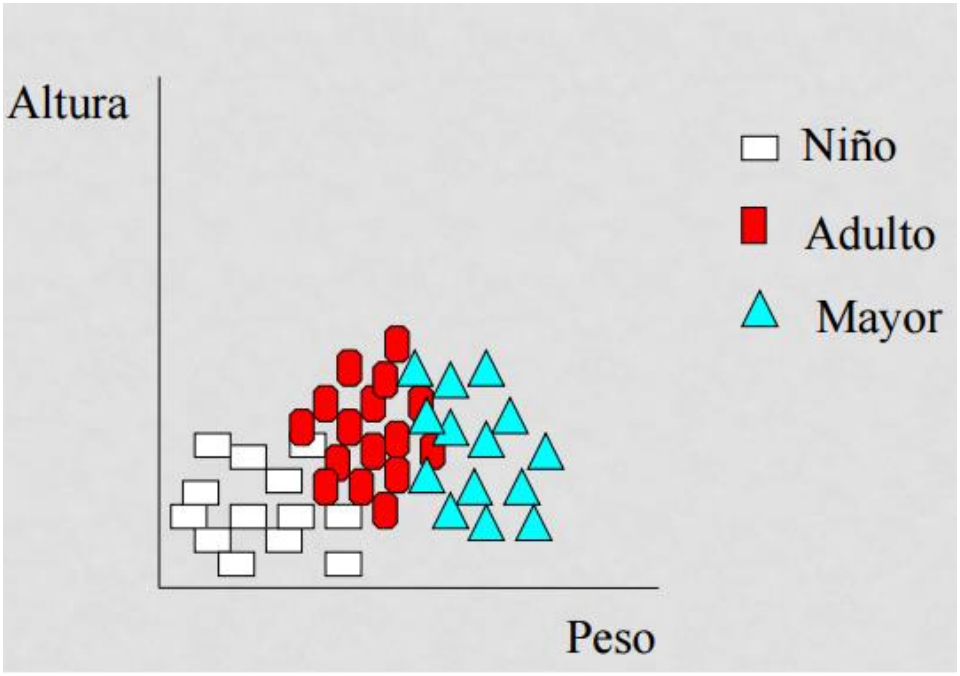

Figura 2.7 Datos de ejemplo KNN.

Una vez que se tienen los ejemplos almacenados, y elegido el número de vecinos que tomarán partido en la clasificación, se pasa al siguiente paso. Se elige un dato nuevo y se introduce en el sistema. Dependiendo del número de vecinos que entren en juego, se compararán la clase a la que pertenecen los k vecinos más cercanos. La clase mayoritaria entre los vecinos que entren en juego, se establecerá como salida.

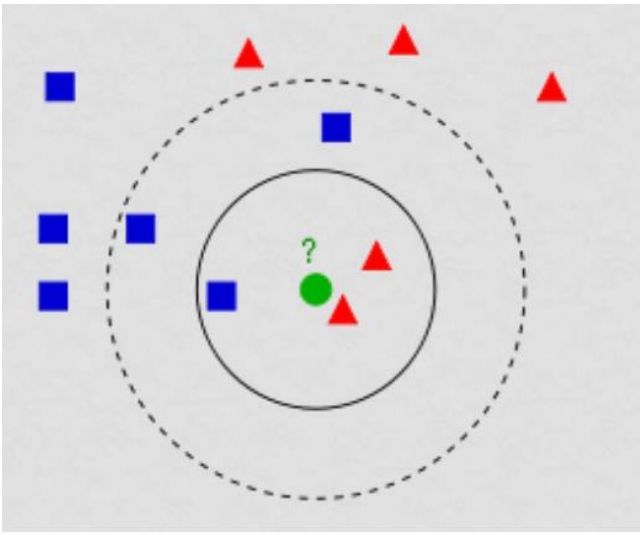

Figura 2.8 Clasificación de un dato con KNN.

En la figura 2.8 se observa que se ha introducido un nuevo dato en el sistema y que se desconoce la clase a la que pertenece. En el ejemplo si para clasificar el nuevo dato (punto verde) se utiliza un k=1 pertenecerá a la misma clase que pertenezcan los triángulos rojos. Si en vez de 1, k vale 5, a la clase de ese dato será asignada la clase a la que pertenezcan los cuadrados azules, puesto que en los 5 vecinos más cercanos, la clase de los cuadrados azules es mayoritaria que la de los triángulos rojos.

2.4.3 PCA: Principal Component Analysis.

El método de PCA se basa en aplicar una transformación lineal a los datos para poder seleccionar a los más importantes o relevantes.

Para ello, el método comienza tomando los atributos conocidos y realizando una combinación lineal para crear nuevos atributos en base a los anteriores. Una vez creados los nuevos atributos, son ordenados de más a menos relevante, para que los que sean más importantes queden en las primeras posiciones. El principal objetivo de utilizar este método es que se puede reducir la dimensionalidad del problema si en lugar de tomar todos los atributos, se selecciona un número inferior formado por los que contienen información fundamental.

Lo que consigue PCA es que al aplicar la transformación lineal a los atributos, cambia el sistema de coordenadas en el que están distribuidos los datos originales y fija en uno de los ejes el primer componente principal, que es identificado gracias a tener la varianza de mayor tamaño. En el segundo eje, la segunda varianza de mayor tamaño y así de manera sucesiva.

Lo que hace que el método sea interesante de utilizar, es que a la hora de reducir la dimensionalidad de los datos, mantiene las características de los datos que son más críticas para su varianza, y por tanto se queda con aquellos componentes que tienen la información más relevante.

El método relaja todo su funcionamiento en las covarianzas, ya que quiere reducir un grupo de datos con una dimensión en otro conjunto de datos de menor dimensión sin perder información. Los datos que se van a utilizar deben estar centrados a media 0, y preferiblemente autoescalados siguiendo la siguiente ecuación:

$$X = \sum_{a=1}^{l} t_a p_a^T + E$$

Ecuación 2.11 Ecuación para el análisis de datos PCA.

Donde $'t_a'$ recibe el nombre de 'scores' y almacena la información sobre la relación entre unas muestras y otras. Posee la característica de ser ortogonales. $'p_a'$ recibe el nombre de 'loadings' y contiene la información de la relación que existe entre las variables que entran en juego, posee la característica de ser ortonormales. Como no se eligen a todos los atributos, se produce un error 'E' que se añade a la matriz.

Una vez se tiene la matriz de covarianza, se pasa a descomponer la matriz en vectores propios siguiendo la siguiente ecuación:

$$\begin{aligned} &\operatorname{cov}(X) = \frac{X^T X}{n-1} \\ &\underset{m}{\operatorname{cov}}(X)\ \mathbf{p}_a = \lambda_a\ \mathbf{p}_a \\ &\sum_{a=1} \lambda_a = 1 \end{aligned}$$

Ecuación 2.12 Descomposición de la matriz en vectores propios.

Donde λ_a es el valor propio de p_a y $t_a = X\, p_a$. Por lo que t_a son las proyecciones de la matriz en p_a. λ_a recoge la información que cada uno de los componentes principales representa. Al estar ordenados, el primer componente es el más importante de todos, mientras que el último será el menos relevante.

De una manera gráfica se puede mostrar lo que se pretende y cómo funciona PCA.

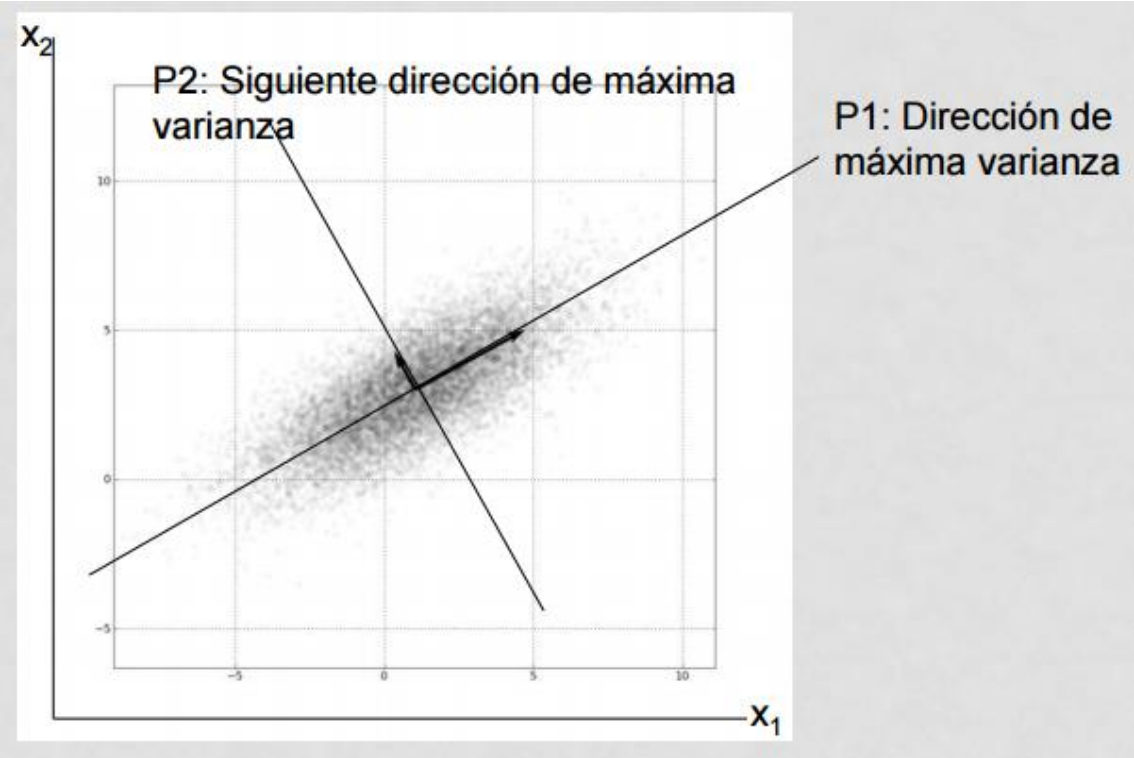

Figura 2.9 Creación de nuevos atributos PCA.

En la figura 2.9 se observa una nube de puntos que representa todos los datos de entrada, tras realizar el cálculo se obtiene que la máxima varianza se sitúa en la parte larga de la especie de espiral que forman los puntos, por lo tanto se fija en la dirección de la máxima varianza uno de los ejes. A continuación se observa que la siguiente máxima varianza se sitúa en la parte estrecha de la espiral y por tanto se toma la dirección de dicha varianza y se fija el siguiente eje. Con ello lo que se consigue es la transformación lineal de los datos y que así roten hasta colocarse en un nuevo eje de coordenadas.

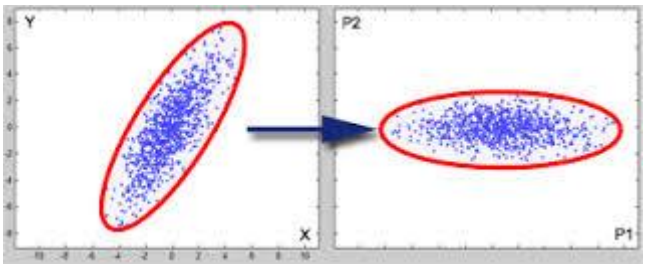

Figura 2.10 Rotación de los datos PCA.

Como se observa en la figura 2.10 los datos han sido rotados de tal manera que las dos máximas varianza están representadas en X e Y del eje de coordenadas.

Como cualquier método, PCA también tiene inconvenientes. Al ser un filtro no supervisado, no garantiza que los atributos que genera sean los que discriminan mejor a las clases. Por otro lado, en el momento que entran en juego muchos atributos, su comportamiento es lento.

2.4.4 Herramientas.

2.4.4.1 Matlab.

Para la realización de la programación necesaria para este estudio se ha utilizado el lenguaje de alto nivel y entorno interactivo Matlab. Es un programa que es utilizado por un gran número de científicos e ingenieros alrededor del mundo. La versatilidad que tiene lo convierte en una herramienta muy potente y que puede ser utilizada en multitud de proyectos.

Matlab cuenta con un gran número de 'toolbox', y entre ellas se encuentra una específica para las redes neuronales y por tanto hace que sea la herramienta perfecta por su potencia y especificación para realizar el estudio.

Esta extensión del programa se llama 'Neural Nework Toolbox'y facilita funciones y aplicaciones para crear sistemas complejos no lineales que son aquellos modelos que no son fáciles de implementar con una única ecuación. Es capaz de trabajar con distintos tipos de aprendizaje tanto supervisado como no supervisado, es capaz de poder diseñar, entrenar simular y visualizar redes de neuronas y por tanto puede ser usado para para ajuste de datos, reconocimiento de patrones, agrupación, predicciones de series temporales y el modelado de sistemas dinámicos y de control.

2.4.4.2 Microsoft Excel.

Excel es una herramienta muy conocida distribuida por Microsoft que permite crear tablas e insertar en ellas cálculos matemáticos o fórmulas para trabajar con un gran número de datos a la vez.

Se ha utilizado Excel en este trabajo para poder normalizar, aleatorizar y separar los datos de entrada de los diferentes casos utilizados en el estudio. Los datos contienen información real que no puede ser utilizada directamente en los métodos utilizados. Es necesario aleatorizar todos los datos para evitar posibles patrones en ellos y que el estudio se realice de una mejor manera. También se utiliza este programa para normalizar dichos datos y para separar en varios conjuntos el bloque de datos original y poder conseguir de esta manera conjuntos de entrenamiento y conjuntos de test.

Capítulo 3: Diseño del sistema

3. Diseño del sistema.

En este apartado se explica el diseño adoptado y se explica de qué manera se ha creado el programa utilizado para recoger los datos mediante los cuales se ha realizado el estudio de reducción de dimensionalidad.

Los diferentes pasos que hay que realizar para llevar a cabo este estudio son los siguientes:

- Realizar una reducción de dimensionalidad utilizando una red de neuronas.
- Extraer los atributos desde la capa oculta de la red de neuronas.
- Aplicar KNN a los atributos recogidos.
- Realizar una reducción de dimensionalidad utilizando el método de PCA para posteriormente aplicar a los nuevos datos KNN.
- Aplicar KNN a los datos originales.
- Utilizar una red de neuronas con los datos originales
- Utilizar los datos recogidos en los apartados anteriores para realizar un estudio sobre el comportamiento.

Como se ha explicado en el apartado 2.4.4.1 se ha tomado la decisión de utilizar el lenguaje y entorno de Matlab por tener una caja de herramientas amplia en lo que se refiere a redes de neuronas, capaz de poder crear, entrenar y visualizar de manera gráfica una red de neuronas, por ser un lenguaje de alto nivel muy potente y por su gran poder de computación.

Para crear el sistema se ha tenido que implementar varias funciones y métodos que se pasan a enumerar a continuación:

- Obtener las activaciones de la capa oculta.
- Aplicar KNN a las activaciones de la capa oculta.
- Aplicar KNN a los datos originales.
- Aplicar PCA a los datos originales y aplicar KNN.

Antes de comenzar a explicar las diferentes funciones se debe hacer un comentario general en cuanto al tratamiento de los datos. Se deben preparar o preprocesar para poder trabajar con ellos. Es decir, los datos tienen que estar normalizados, aleatorizados y separados en dos bloques, uno que será el conjunto de entrenamiento para generar los modelos y el otro será el conjunto de test para evaluar dichos modelos.

3.1 Introducción general.

Se ha decidido apoyar todas las explicaciones de los apartados siguientes añadiendo un gráfico en el que se muestre de manera general la idea del trabajo que se va a realizar.

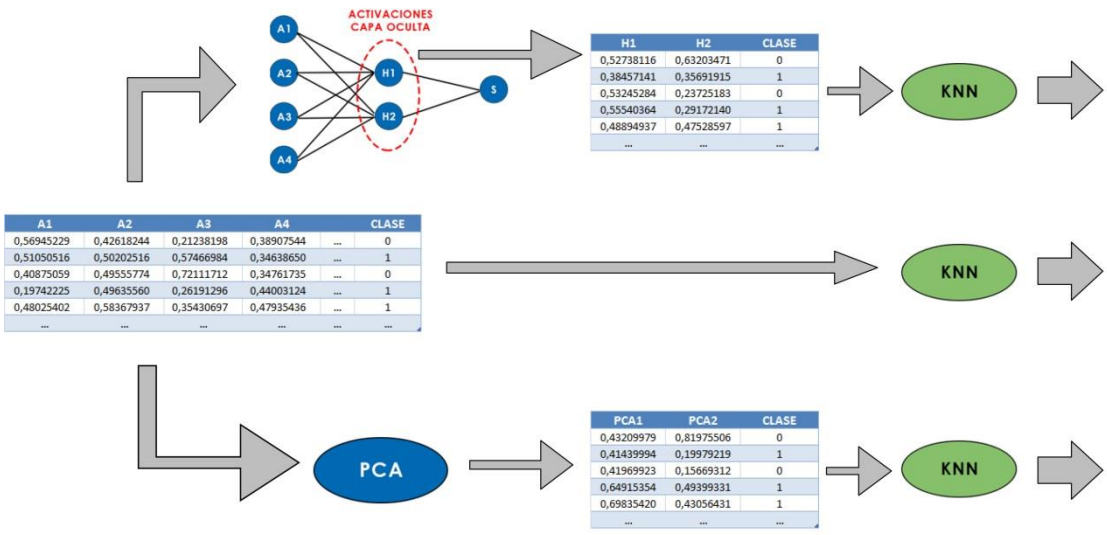

Figura 3.1 Esquema del sistema y experimentación realizada.

Como se puede observar en la figura 3.1 se parte siempre de los datos que forman el dominio; dichos datos ya se encuentran normalizados, aleatorizados y separados en dos conjuntos: el conjunto de entrenamiento y el conjunto de test. A continuación esos mismos datos pasarán por tres procesos totalmente diferentes:

En orden descendente del gráfico mostrado en la figura 3.1 se pasan a explicar los métodos. En el primero de los casos se utiliza una red de neuronas en la cual los datos se introducen en la capa de entrada, la red realiza su operaciones normales, pero en vez de recoger la salida facilitada por la red, que sería el comportamiento normal, nos detenemos a recoger las activaciones de la capa oculta de la red de neuronas. A dichas activaciones se las asigna la clase a la que pertenecen y seguidamente se realiza una clasificación mediante KNN utilizando los nuevos datos reducidos. En el caso de la figura se comienzan con cuatro datos y se pasa a realizar la clasificación únicamente con dos. A este método lo abreviaremos como NN+KNN.

El siguiente proceso que está representado gráficamente es en el que se toman los datos originales y se aplica una clasificación KNN directamente a ellos para saber cómo de bien el clasificador es capaz de trabajar.

En último lugar aparece el proceso que consiste en que a los datos originales se les aplique el método PCA. PCA devuelve los componentes ordenados de mayor a menor importancia. Lo siguiente es elegir cuántos mejores componentes utilizar, se recogen todos ellos y se les asigna la clase a la que pertenecen. Una vez creado los nuevos datos, al igual que en el resto de procesos se aplica KNN para realizar una clasificación y comprobar si la reducción produce resultados satisfactorios. A este método lo abreviaremos como PCA+KNN.

3.2 Obtener las activaciones de la capa oculta.

El objetivo final de crear una función capaz de obtener las activaciones de la capa oculta es realizar una transformación de las N dimensiones de los datos de entrada en las M dimensiones de las neuronas que forman la capa oculta de la red. Realizando dicha transformación, conseguiremos reducir la dimensionalidad de los datos. Por supuesto M siempre debe ser menor que N para que se pueda conseguir la reducción de dimensionalidad.

Una vez los datos están preprocesados, se pasa a cargar en la función los dos conjuntos de datos, el bloque de entrenamiento y el bloque de test. Tanto del bloque de entrenamiento como del bloque de test, es necesario que la función separe los atributos de la clase a la que pertenecen para poder realizar las siguientes operaciones.

A continuación se debe decidir el número de neuronas que van a componer la capa oculta de la red, para seguidamente crear una red de neuronas con dicho número de neuronas en ella.

El siguiente paso es entrenar la red, se le asigna un número alto de iteraciones para que la red se entrene correctamente y se le asigna un número máximo de fallos de entrenamiento que la red puede permitirse antes de detenerse.

Una vez la red está entrenada se pasa a guardar las salidas de las clases para poder calcular en siguientes pasos el error cometido por la red y saber cómo de bien ésta se ha entrenado. Lo siguiente es almacenar en un array las activaciones de las neuronas de la capa oculta, para ello se debe recorrer todas las neuronas de la capa oculta e ir calculando su activación.

En este punto la red está entrenada y se tiene almacenado en un array las activaciones de las neuronas, por tanto el siguiente paso es probar la red. Para ello a la red entrenada, se le pasa el conjunto de test y se simula su ejecución. Al igual que en el entrenamiento, se calcula el error cometido por la red para el conjunto de test y se calculan y almacenan las activaciones de las neuronas de la capa oculta.

Por último la función llama a otra función que se encarga de aplicar KNN a las activaciones. Para ello facilita por parámetro los vectores que contienen las activaciones de las neuronas de entrenamiento y de test.

3.3 Aplicar KNN a las activaciones de la capa oculta.

Esta función tiene como objetivo aplicar KNN a las activaciones de las neuronas que forman parte de la capa oculta de la red, es decir, a los datos transformados, y calcular el error cometido.

Para ello, la función recibe por parámetro las activaciones de las neuronas de la capa oculta de la red, obtenidas con el conjunto de entrenamiento y de test. Por otro lado se cargan los conjuntos de datos originales con los que se están trabajando para poder separarlos en dos bloques, el bloque de los parámetros y el bloque de la clase a la que pertenecen. Esta separación se aplica tanto al conjunto de entrenamiento como al conjunto de test.

Una vez los datos han sido separados en los diferentes bloques se pasa a crear el modelo de clasificación KNN. Lo que se pretende es aplicar una clasificación KNN a las activaciones de la capa oculta de la red para ver si se ha reducido los datos sin perder información importante, y así poder comprobar si es posible clasificar de manera correcta los nuevos datos. Para ello se crea un modelo a partir de las activaciones. Como resultado de esas activaciones, se deben obtener las salidas de la clase que pertenecen los datos de los que proceden dichas activaciones, por tanto se toma de los datos originales las salidas deseadas.

Para crear el modelo se debe indicar el número de vecinos que se desean utilizar en la clasificación.

Con el modelo creado a partir de las activaciones obtenidas en el entrenamiento, se predicen las salidas para los nuevos datos utilizando las activaciones recogidas del conjunto de test. A continuación se calcula el error cometido comparando las salidas ofrecidas por el modelo que ha utilizado los datos de las activaciones del conjunto de test con las salidas deseadas que son las salidas a la que pertenecen los datos que han creado las activaciones del conjunto de test.

El proceso se debe repetir para cada número de vecinos diferente que se quiera usar a la hora de aplicar la clasificación KNN.

3.4 Aplicar KNN a los datos originales.

Esta función se encarga de aplicar el clasificador KNN a los datos originales. Esto se realiza para poder comparar el comportamiento del método con los datos originales y con los datos reduciendo su dimensionalidad y así poder analizar su comportamiento.

Para ello en la función se carga el conjunto de entrenamiento y de test, y al igual que en las funciones anteriores se debe separar en dos bloques los parámetros de los datos y la clase a la que pertenecen.

A continuación se debe crear un modelo de clasificación KNN en el cual los datos para crearlo serán el conjunto de datos de entrenamiento y como salida deseadas las salidas que se han separado de este mismo conjunto de entrenamiento. Con ello el modelo ya tiene las clases a la que pueden pertenecer los nuevos datos que se introduzcan en la fase de test.

Para la fase de test se desea que el modelo clasifique de manera correcta el mayor número de datos utilizando los atributos del conjunto de test, para poder asignarle a cada uno la clase a la que pertenecen. Por eso se le facilita al modelo creado anteriormente el conjunto de test.

El clasificador predice las clases a las que deberían pertenecer los nuevos datos, éstas se almacenan y son comparadas con las salidas deseadas que son las salidas que se han separado al comienzo de la función del conjunto de entrenamiento. De esta comparación se obtiene el porcentaje de fallos que se ha producido.

3.5 Aplicar PCA a los datos originales y aplicar KNN.

Esta función tiene que realizar dos tareas, por un lado debe realizar la reducción de atributos de los datos originales utilizando el método de PCA y seleccionar de entre ellos a los mejores, y por otro lado debe realizar una clasificación KNN con estos nuevos atributos.

Se comienza cargando en la función el conjunto de entrenamiento y el conjunto de test y se separa como en todos los casos anteriores el conjunto propio de datos, de la clase a la que pertenecen. Este proceso se realiza tanto en el bloque de entrenamiento como en el grupo de test.

A continuación se aplica PCA al conjunto de datos, para ello Matlab tiene su función propia, PCA, que aplica el método a los datos que son facilitados. En este caso se pasan los datos de entrenamiento y la función devuelve los atributos ordenados de mayor a menor relevancia. A continuación se selecciona el número de datos que se quieren utilizar, descartando el resto.

El siguiente paso es generar un modelo de clasificación KNN en el cual entran en juego los mejores componentes seleccionados en el método PCA, las salidas del entrenamiento a la que pertenecen dichos componentes y el número de vecinos que se quiere utilizar en la clasificación.

Una vez que se tiene el modelo creado, se aplica PCA a los datos del conjunto de test para seleccionar entre ellos a los mejores atributos de igual manera que se ha actuado con los datos del conjunto de entrenamiento, es decir, se le facilita a la función propia de Matlab, PCA,

el conjunto de test, y ésta devuelve los atributos ordenados de mayor a menor importancia, de los cuales se selecciona el mismo número de atributos que han sido seleccionados del conjunto de entrenamiento.

Con los atributos ya seleccionados se facilita a KNN el conjunto de atributos del grupo de test para que sean clasificados utilizando el modelo anteriormente creado con los atributos del conjunto de entrenamiento. A continuación se comparan las salidas reales que se recogen de la clasificación KNN y se comparan con las salidas deseadas que se han separado al comienzo de la función para calcular el porcentaje de error que se ha cometido.

3.6 Entorno tecnológico.

A continuación se expone la infraestructura técnica que se ha utilizado para el desarrollo del código utilizado para el estudio del trabajo.

3.6.1 Hardware.

Para el desarrollo de la correcta codificación y ejecución del programa se ha utilizado un equipo con las siguientes características:

- Procesador Intel Core2 Quad a 2.83 GHz
- Memoria RAM de 8 GB
- 128 GB memoria de disco duro
- Unidad de CD-ROM
- Monitor
- Ratón y teclado
- Equipo conectado a una red.

3.6.2 Software.

En cuanto a software se han utilizado varios programas. Al elegirse Matlab como lenguaje de alto nivel a la hora de programar el código, se utilizó el entorno de Matlab R2014a junto con la caja de herramientas preparada para redes de neuronas (Neural network toolbox).

Para el trabajo con los datos de entrada, que debían ser normalizados, aleatorizados y separados en subgrupos, se ha utilizado Microsoft Excel 2010 que pertenece al paquete de gestión ofimática de Microsoft Office 2010.

Para la redacción de la documentación del trabajo de fin de grado, se ha utilizado otro programa del paquete de Microsoft Office 2010: Microsoft Word 2010.

Todos los programas anteriormente citados y utilizados necesitan de un sistema operativo capaz de gestionarlo, en este caso se ha utilizado Windows 7 de 64 bits.

3.6 Alternativa de diseño.

Por último se justifica las decisiones de diseño que se han tomado y las razones por las que han sido tomadas.

Lenguaje de alto nivel: Al comienzo del trabajo de fin de grado se estuvieron analizando los posibles lenguajes a escoger. De todos los candidatos, la elección iba a estar entre dos: Matlab o R.

A primera vista son bastante similares, ambos tienen acceso a funciones matemáticas, un lenguaje propio, estadísticas y una comunidad de usuarios, sin embargo si centramos un poco más la visión se observa que Matlab aporta ciertas mejoras que hacen que la balanza se torne en su favor.

Matlab proporciona un entorno de desarrollo de alta productividad en la que están incluidas amplias cajas de herramientas específicas con lo que es fácil que la productividad sea mayor utilizando Matlab y no R. Por ejemplo para este trabajo de fin de grado ha sido fundamental la caja de herramientas que corresponde al aprendizaje automático.

También se conoce por estudios realizados que Matlab es más rápido que R cuando se ejecutan las mismas pruebas. Pero la razón definitiva es la documentación que Matlab facilita al usuario en la cual permite buscar información en línea o dentro del escritorio de Matlab y que está formada por cientos de ejemplos con código.

Por todos estos aspectos se decidió escoger Matlab en lugar de R a la hora de desarrollar la aplicación que obtiene los resultados.

Herramientas ofimáticas: En cuanto al trabajo que se tiene que realizar con las bases de datos se ha elegido Microsoft Excel debido a que es una aplicación potente que puede realizar los cálculos necesarios de una manera eficaz y su reputación está más que probada debido a la cantidad de tiempo que lleva en el mercado. Además de esto, es una herramienta que resulta muy útil ya que aparte de los cálculos que es capaz de hacer con grandes bases de datos, se puede utilizar en multitud de ocasiones a la hora de crear tablas para la documentación.

Además de lo anterior, debido a que la Universidad posee una licencia, y por tanto no supone ningún gasto, es el mejor candidato para realizar dichas tareas en lugar de otros productos de licencia gratuita.

Capítulo 4: Estudio realizado

4. Estudio realizado.

En este apartado se explica el estudio que se ha realizado y los resultados obtenidos, analizando los datos y exponiendo las conclusiones que se pueden recoger después de aplicar las técnicas de reducción de dimensionalidad, así como la metodología utilizada.

4.1 Metodología utilizada.

Es importante, a la hora de realizar un estudio de investigación, concretar la metodología que se va a utilizar en el momento de obtener los datos, para que se adquieran de manera que todas las pruebas realizadas, se puedan comparar y así poder saber el comportamiento general del sistema.

En este caso se van a utilizar diferentes técnicas para realizar la reducción de dimensionalidad, en las que entran en juego diferentes variables que hacen que el sistema sea capaz de clasificar de mejor o peor manera.

Hay que recordar que la idea general del trabajo es realizar una investigación mediante la cual se descubrirá si es posible reducir los atributos de un problema utilizando diferentes métodos, y comprobar el funcionamiento del sistema una vez fuera de juego dichos parámetros.

Por tanto, se ha decidido que para las pruebas, la reducción de atributos será del 50% en adelante, es decir, que las pruebas que se realizarán utilizarán el 50 % o menos de los atributos originales del dominio. De esta manera, al ser una prueba agresiva, si el sistema recoge buenos resultados realizar la reducción de atributos estaría justificado, puesto que se quitaría mucho peso de computación y por tanto el tiempo de ejecución se vería reducido de forma considerable.

Una vez decidido que como mucho se van a utilizar hasta el 50 % de los atributos que tenga el conjunto original, se deben repartir las pruebas a realizar. Para ello se ha decidido que

se van a utilizar siete arquitecturas generales, en las cuales se variarán pequeños parámetros para comprobar su comportamiento final.

Es decir, si un dominio tiene setenta atributos de entrada, se realizarán siete arquitecturas en las que la variable principal que afecta de mayor manera al comportamiento de ellas, irá variando de una manera progresiva y constante. Por seguir con el ejemplo del dominio que contiene 70 atributos de entrada, las componentes principales de los diferentes métodos deben ser: 5, 10, 15, 20, 25, 30 y 35.

En el caso de que se utilice una red de neuronas, estos números representarán el número de neuronas que entran en juego en la capa oculta, en caso de que el método sea PCA, representarán el número de mejores componentes que entrarán en juego en la clasificación.

Como una vez terminada la reducción de dimensionalidad se utiliza un clasificador para conocer cómo de correcta y productiva ha sido dicha reducción, se debe establecer la metodología que se utilizará para el clasificador.

Uno de los componentes más importantes que entran en juego al utilizar el clasificador KNN, es el número de vecinos que entran en juego a la hora de decidir qué salida generarán las nuevas instancias del conjunto de prueba. La metodología utilizada para este paso ha sido realizar para todas las arquitecturas cuatro pruebas en las que entrarán en juego 1, 3, 5 y 7 vecinos.

Los números elegidos para los vecinos son impares para que de esta manera no se produzcan empates que hagan que el clasificador no sepa relacionar una entrada con una salida concreta.

Una vez que se han expuesto las pruebas que se deben de realizar en cada uno de los diferentes procesos, se establece la metodología general. Constará de los siguientes pasos.

(1) Preprocesado de los datos (aleatorización y normalización).
(2) Separación de la base de datos en conjunto de entrenamiento y conjunto de test.
(3) Ejecución de la red con los datos de entrenamiento.
(4) Extracción de los datos de la capa oculta y prueba con KNN y datos de test.
(5) Aplicación de PCA y prueba con KNN y datos de test.
(6) Aplicación de KNN con datos de test.
(7) Comparación de KNN, NN+KNN y NN+PCA.

(1) y (2). A cada conjunto de datos hay que separarlo en otros dos subconjuntos: conjunto de entrenamiento y conjunto de test. Para ello, anteriormente, se debe realizar un preprocesado de los datos.

Comencemos explicando el preprocesado de los datos. Todos los dominios que se van a utilizar en el este estudio (excepto uno), proceden del mundo real y por tanto en la base de datos están almacenados atributos de muy distintas características. Por lo que se deben unificar estos atributos para poder trabajar correctamente con ellos. Para unificar los atributos de un conjunto de datos, se deben realizar distintas operaciones. El primer paso que se va a realizar es aleatorizar los datos. Los datos pueden estar almacenados en sus bases de datos creando ciertos patrones que no son útiles y que pueden modificar el resultado final. Para aleatorizar los datos nos ayudaremos de Excel. En la base de datos original se añadirá una nueva columna con números aleatorios, seguidamente se organizará todas las columnas tomando como referencia el orden ascendente de los números aleatorios creados en la nueva columna. De esta manera, como resultado final, obtenemos los datos originales pero su posición en la base de datos se ha aleatorizado.

A continuación una vez aleatorizados los datos, se pasa a normalizarlos. Es importante a la hora de trabajar con los datos que éstos estén comprendidos en un rango concreto, por lo tanto se decide normalizar todas las bases de datos que se van a utilizar entre 0 y 1. Para realizar esta normalización se utilizará Excel y se aplicará la siguiente fórmula a todos los datos:

$$Norm = \frac{Entrada - MIN(TotalEntradas)}{MAX(TotalEntradas) - MIN\ (TotalEntradas)}$$

Ecuación 4.1 Normalización establecida.

Una vez los datos están aleatorizados y normalizados, se puede pasar al siguiente paso: separar el dominio en conjunto de entrenamiento y conjunto de test. Para este paso se ha decidido que los conjuntos de entrenamiento estén formados por el 70 % del total de los datos, quedando el otro 30 % restante como conjunto de test.

(3) y (4). Ejecución de la red con los datos de entrenamiento y extracción de información de la capa oculta de la red, realizando una prueba con KNN y los datos de test. Una vez que los datos están preparados se debe introducir en una red de neuronas el conjunto

de entrenamiento, para seguidamente, poder extraer de la capa oculta de dicha red el nuevo conjunto de datos creado por la transformación y reducción de los originales. A continuación, con los nuevos atributos, se crea un modelo con KNN al que le pide que realice una clasificación del conjunto de test (formado por las activaciones de la capa oculta de la red para el conjunto de test) y así poder calcular el porcentaje de error que comete.

(5) Para aplicar PCA y poder realizar una prueba con KNN y los datos de test, se debe reducir la dimensionalidad del conjunto de entrenamiento y de test mediante PCA y seleccionar los X mejores atributos. Con la reducción de dimensionalidad que PCA realiza al conjunto de entrenamiento, se crea un modelo con KNN al que se le pide que clasifique el conjunto de test que ha sido reducido por PCA, (y del que se han seleccionado el mismo número de mejores atributos que para el conjunto de entrenamiento) y así poder calcular el porcentaje de error que comete.

(6) Aplicar KNN con los datos de test. Para este paso se toma el conjunto de entrenamiento y de test del dominio al que se quiere someter a prueba. Se toma el conjunto de entrenamiento y se crea un modelo KNN al que seguidamente se pide que clasifique el conjunto de test para calcular el porcentaje de error que comete.

(7) Por último se realiza una comparación general de todos los errores obtenidos en los pasos anteriores: El porcentaje de error cometido por KNN, el porcentaje de error cometido por NN+KNN y el porcentaje de error cometido por PCA+KNN.

4.2 Conjunto de datos.

Los conjuntos de datos que se han utilizado para la realización del estudio han sido recogidos, si no la totalidad, si la mayoría de ellos del repositorio alojado en: http://sci2s.ugr.es/keel/category.php?cat=clas&order=featR#sub2.

En concreto los dominios utilizados han sido:

- Spambase
- German
- Estate
- Splice
- Dominio con los datos girados

4.2.1 Spambase.

Spambase es una base de datos obtenida de datos reales que contiene información acerca de 4597 mensajes de e-mail. El objetivo que se pretende con este dominio es determinar si los e-mails son correos tipo spam o no. En su mayoría los atributos indican si una palabra o carácter se repite de manera frecuente en el email. En total hay 57 atributos acerca de los correos:

- 48 de ellos son atributos continuos reales que indican el porcentaje de palabas que coinciden con la palabra buscada. Se tomará como palabra a la cadena de caracteres alfanuméricos delimitados por caracteres no alfanuméricos o por el final de una cadena.
- 6 son atributos reales continuos de tipo carácter que es el porcentaje de caracteres que tiene el e-mail que coinciden con el carácter buscado.
- 1 atributo real continuo que almacena la longitud media ininterrumpida separada por mayúsculas.
- 1 atributo entero continuo que almacena la longitud máxima ininterrumpida separada por mayúsculas.
- 1 atributo entero continuo que almacena el número total de letras mayúsculas que contiene el e-mail.

4.2.2 German.

German es una base de datos obtenida de datos reales que contiene un total de 1000 instancias con datos propios de los usuarios y de sus cuentas bancarias. El objetivo es que conociendo estos datos se pueda clasificar a un usuario como buen cliente o mal cliente.

En total tiene 20 características que quedan enumeradas en la siguiente lista:

- Estado de la cuenta
- Duración en meses
- Historia de crédito
- Propósito de la cuenta
- Crédito

- Cuenta de ahorros
- Empleado desde
- Tasa de los tramos de préstamo
- Estado y sexo
- Aval
- Tiempo de Residencia
- Propiedad
- Edad
- Viviendas
- Número de créditos
- Trabajo
- Número de personas en el núcleo familiar
- Teléfono
- Trabajador extranjero

4.2.3 Dominio con los datos girados.

Este dominio es un dominio especial, se trata de un dominio artificial. El dominio tiene forma de una elipse de puntos, está formado por dos clases y tiene dos atributos, sin embargo se le han añadido 8 atributos más con valores entre 0 y 1 asignados de manera aleatoria y uniforme. Como resultado se obtiene un dominio de 10 dimensiones, cuyos datos han sido rotados aleatoriamente, creando finalmente un domino formado por 10 atributos de entrada y uno de salida.

Utilizar este dominio en el estudio es interesante puesto que se conoce a priori, de manera aproximada los resultados que se deben obtener. Al utilizar la red de neuronas, ésta debe ser capaz de eliminar los atributos innecesarios y por tanto pasarle a KNN únicamente aquellos atributos que son importantes a la hora de clasificar.

A continuación se muestra una imagen del dominio en 2D, antes de añadirle los 8 atributos aleatorios:

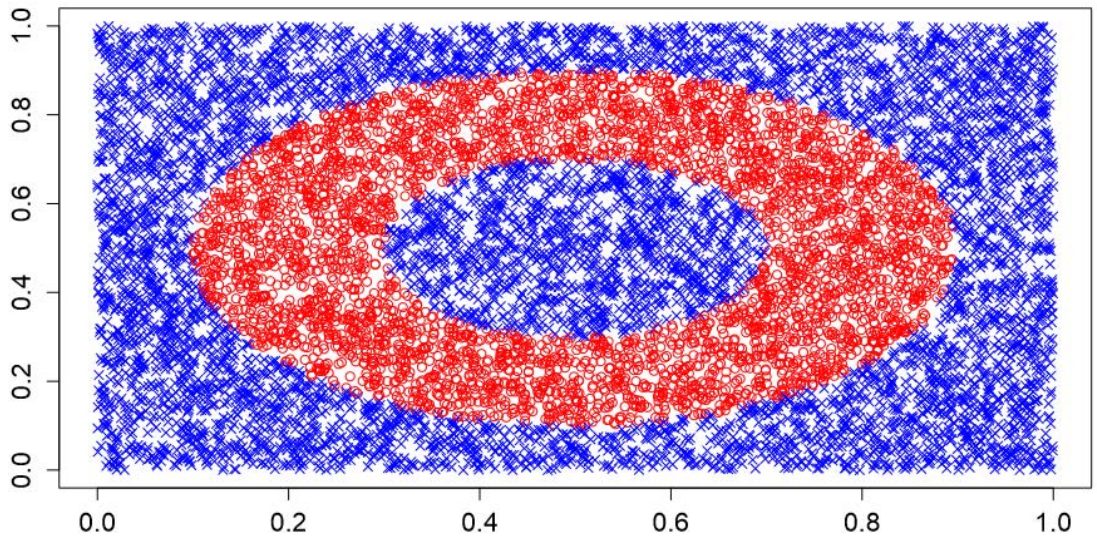

Figura 4.1 Dominio generado de manera artificial.

4.2.4 Estate.

La descripción de este dominio no está accesible. Pero ha sido utilizado en otros trabajos como este: "Cieslak, D. A., Hoens, T. R., Chawla, N. V., & Kegelmeyer, W. P. (2012). Hellinger distance decision trees are robust and skew-insensitive. Data Mining and Knowledge Discovery, 24(1), 136-158."

4.2.5 Splice.

Splice es una base de datos que proviene del repositorio UCI de base de datos de aprendizaje automático. El objetivo de este conjunto de datos es reconocer dos tipos de nudos de empalme en las secuencias de ADN. Las dos uniones son las dos clases que tiene el problema exón/intrón por un lado y intrón/exón por otro. Un nudo de empalme es el lugar en una cadena de ADN donde se retira el ADN superfluo en el momento de creación de una proteína.

Intrón es la parte que se refiere a la porción de secuencia que se une a la cadena, mientras que el exón es la parte de la secuencia que se retira.

El dominio está formado por un total de 1000 instancias, obtenidas del mundo real que contiene 60 atributos con información acerca de los nudos de enlace de la cadena de ADN.

4.3 Pruebas realizadas.

A continuación se muestran los resultados obtenidos del estudio realizado, en los cuales entran en juego tanto el conjunto de datos como la metodología anteriormente explicada.

4.3.1 Resultados obtenidos del conjunto de datos artificial girado.

En este apartado se encuentran los resultados obtenidos para poder realizar el estudio de reducción de dimensionalidad utilizando la base de datos girada de manera artificial.

4.3.1.1 Error producido por una red de neuronas.

La tabla 4.1 contiene los resultados obtenidos al aplicar una red de neuronas en la que se han ido variando el número de neuronas que forman parte de la capa oculta.

Nº Neuronas	% de Error Test
1	39.4535
2	35.8880
3	36.3545
4	38.8204
5	21.3595
6	15.1283
7	7.6974

Tabla 4.1 Error de test utilizando una red de neuronas (Girado).

Se observa que el menor porcentaje de error obtenido para el conjunto de test formado por los datos artificiales es un 7.6974% que se capta cuando se utilizan 7 neuronas en la capa oculta de la red.

4.3.1.2 Error producido por NN+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN, al cual en vez de facilitarle los datos originales de la base de datos artificial, se le facilitan los datos transformados en los cuales se ha utilizado una red de neuronas para reducir la dimensionalidad de los mismos.

Nº Vecinos / Nº Neuronas	1	3	5	7
1	45.3182	44.9184	36.7877	43.9853
2	39.2203	37.9540	37.1876	36.5545
3	37.1543	36.8877	35.5215	34.9550
4	39.2536	37.1876	35.7881	35.2216
5	33.8554	35.8581	35.6548	35.0883
6	4.9317	4.4319	4.4985	4.2652
7	2.0993	1.9327	2.0660	2.0660

Tabla 4.2 Error de test de NN+KNN (Girado).

Como se observa en la tabla 4.2, el menor porcentaje de error que se comete es el 1.9327%, recogido en la prueba NN+KNN en la que se reduce a 7 atributos utilizando la capa oculta de la red de neuronas y se utiliza para el clasificador los 3 vecinos más cercanos. Además se puede observar que hacen falta al menos 6 componentes para recoger buenos resultados y que el número de vecinos no influye demasiado en las pruebas.

4.3.1.3 Error producido por PCA+KNN.

A continuación se muestran los resultados, expresados en porcentaje del error, que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos girada artificialmente, se le facilitan los datos de los N elementos mejores que son decididos utilizando el método de PCA.

Además, para poder realizar de mejor manera la comparación se añade en esta misma tabla los resultados que se obtiene al aplicar KNN directamente a los datos originales en los que entran en juego todos los atributos que forman el conjunto de datos girados artificialmente.

Nº Componentes \ Nº Vecinos	1	3	5	7
Originales (sin reducir)	38.3206	36.7877	37.4875	37.0543
1	47.3176	44.2186	43.8854	43.1190
2	47.0510	45.9513	45.5182	44.5518
3	46.2179	44.9517	45.4515	44.6518
4	47.4842	45.8514	45.6848	43.9520
5	47.5841	46.3512	45.0183	44.8517
6	46.7511	45.6514	45.0183	44.6185
7	46.6178	45.4182	43.2856	42.7857

Tabla 4.3 Error de test de PCA + KNN y KNN a los datos originales (Girado).

Lo que se puede observar de la tabla 4.3 es que el menor error recogido por KNN utilizando los datos facilitados por PCA es de un 42.7857 %, mientras que al aplicar KNN a los datos originales, el menor error es del 37.0543 %.

4.3.1.4 Comparación entre NN+KNN y PCA+KNN.

A continuación se va a representar, con la ayuda de una tabla de resultados, la diferencia entre realizar la reducción de dimensionalidad con una red de neuronas o utilizar el método PCA.

Elementos \ Nº Vecinos	1	3	5	7
1	1,9994	-0,6498	7,0977	-0,8663
2	7,8307	7,9973	8,3306	7,9973
3	9,0636	8,064	9,93	9,6968
4	8,2306	8,6638	9,8967	8,7304
5	13,7287	10,4931	9,3635	9,7634
6	41,8194	41,2195	40,5198	40,3533
7	44,5185	43,4855	41,2196	40,7197

Tabla 4.4 Diferencia entre los errores cometidos por la red de neuronas y PCA (Girado).

En cada casilla se representa la diferencia que existe entre los errores recogidos por KNN al utilizar los datos obtenidos de la reducción de atributos utilizando la red de neuronas o

PCA. Cuanto más alto sea el número, más eficaz será la reducción de atributos mediante la capa oculta de la red de neuronas, comparándolo con la reducción con PCA.

Se puede observar que prácticamente en todos los casos estos valores son positivos, en algunos de los casos con valores altos, lo que significa que compensa reducir la dimensionalidad utilizando una red de neuronas frente a hacerlo con PCA.

4.3.2 Resultados obtenidos del conjunto de datos Spambase.

En este apartado se encuentran los resultados obtenidos para poder realizar el estudio de reducción de dimensionalidad utilizando la base de datos 'SpamBase'.

4.3.2.1 Error producido por una red de neuronas.

Contiene los resultados obtenidos al aplicar una red de neuronas en la que se han ido variando el número de neuronas que forman parte de la capa oculta.

Nº Neuronas	% de Error Test
2	8,0026
6	8,2546
10	7,6184
14	7,9385
18	8,8348
22	7,6184
26	8,3227

Tabla 4.5 Error de test utilizando una red de neuronas (Spambase).

Se observa que el menor porcentaje de error obtenido para el conjunto de test formado por los datos de Spambase es un 7.6184 que se capta en dos ocasiones, cuando el número de neuronas ocultas es igual a 10 y a 22.

4.3.2.2 Error producido por NN+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos Spambase, se le facilitan los datos transformados en los cuales se ha utilizado una red de neuronas para reducir la dimensionalidad de los mismos.

Nº Vecinos / Nº Neuronas	1	3	5	7
2	10,5634	8,7068	8,3867	8,5787
6	8,3227	6,9782	7,0423	6,9782
10	8,2586	7,3624	7,0423	7,6825
14	8,3227	8,0026	7,8105	8,0026
18	9,1549	8,0666	8,3867	7,8754
22	8,3867	8,3227	8,1946	8,2586
26	9,0909	7,2343	7,7465	7,8745

Tabla 4.6 Error de test de NN+ KNN (Spambase).

Como se observa en la tabla 4.6 el menor porcentaje de error que se comete es el 6.9782%, recogido en la prueba que consiste en reducir a 6 elementos utilizando la red de neuronas y utilizar para el clasificador 3 o 7 vecinos.

4.3.2.3 Error producido por PCA+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos Spambase, los datos de los N elementos mejores que son decididos utilizando el método de PCA.

Además para poder realizar de mejor manera la comparación se añade en esta misma tabla los resultados que se obtienen al aplicar KNN directamente a los datos originales en los que entran en juego todos los que forman Spambase.

Nº Componentes \ Nº Vecinos	1	3	5	7
Originales (sin reducir)	10,9475	10,5634	11,0755	11,6517
2	31,306	29,8976	30,2177	30,2177
6	24,1997	23,1754	23,6876	23,7516
10	18,5019	18,1818	18,758	17,9257
14	18,3739	16,6453	17,3496	17,8617
18	15,3009	16,0051	16,5813	16,9654
22	13,6364	13,5723	15,1088	15,4289
26	11,9078	12,2919	12,8681	13,8284

Tabla 4.7 Error de test de PCA + KNN y KNN a los datos originales (Spambase).

Lo que se puede observar de la tabla 4.7 es que el menor error recogido por KNN utilizando los datos facilitados por PCA es de un 11.9078%, mientras que al aplicar KNN a los datos originales, el error se queda en el 10,5634%.

4.3.2.4 Comparación entre NN+KNN y PCA+KNN.

A continuación se va a representar con la ayuda de una tabla de resultados la diferencia entre realizar la reducción de dimensionalidad con una red de neuronas o utilizar el método PCA.

Elementos \ Nº Vecinos	1	3	5	7
2	20,7426	21,1908	21,831	21,639
6	15,877	16,1972	16,6453	16,7734
10	10,2433	10,8194	11,7157	10,2432
14	10,0512	8,6427	9,5391	9,8591
18	6,146	7,9385	8,1946	9,09
22	5,2497	5,2496	6,9142	7,1703
26	2,8169	5,0576	5,1216	5,9539

Tabla 4.8 Diferencia entre los errores cometidos por la red de neuronas y PCA (Spambase).

En cada casilla se representa la diferencia que existe entre los errores recogidos por KNN al utilizar los datos obtenidos de la reducción de atributos utilizando la red de neuronas o PCA. A medida que el número que aparezca sea mayor, significará que la red de neuronas es capaz de reducir atributos de una manera más eficaz que PCA.

4.3.3 Resultados obtenidos del conjunto de datos German.

En este apartado se encuentran los resultados obtenidos para poder realizar el estudio de reducción de dimensionalidad utilizando la base de datos 'German'.

4.3.3.1 Error producido por una red de neuronas.

Contiene los resultados obtenidos al aplicar una red de neuronas en la que se han ido variando el número de neuronas que forman parte de la capa oculta.

Nº Neuronas	% de Error Test
2	31,3333
4	29,6667
6	28,3333
8	27
10	29,6667
12	32,3333
14	26,6667

Tabla 4.9 Error de test utilizando una red de neuronas (German).

Se observa que el menor porcentaje de error obtenido para el conjunto de test formado por los datos de German es un 26.6667 que se capta cuando se utilizan 14 neuronas en la capa oculta de la red.

4.3.3.2 Error producido por NN+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos German, se le facilitan los datos transformados en los cuales se ha utilizado una red de neuronas para reducir la dimensionalidad de los mismos.

Nº Vecinos / Nº Neuronas	1	3	5	7
2	36,6667	31	29	30,3333
4	33,6667	33	28,6667	28,6667
6	31,3333	31,3333	30	30
8	30,6667	28,3333	26,3333	27
10	36,6667	30	28	27,3333
12	29,6667	26	25,3333	25,6667
14	29,6667	28,3333	27,3333	24,3333

Tabla 4.10 Error de test de NN+KNN (German).

Como se observa en la tabla 4.10 el menor porcentaje de error que se comete es el 24.3333%, recogido en la prueba que consiste en reducir a 14 elementos utilizando la red de neuronas y utilizar para el clasificador 7 vecinos.

4.3.3.3 Error producido por PCA+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos German, los datos de los N elementos mejores que son decididos utilizando el método de PCA.

Además para poder realizar de mejor manera la comparación se añade en esta misma tabla lo resultados que se obtiene al aplicar KNN directamente a los datos originales en los que entran en juego todos los que forman German.

Nº Componentes \ Nº Vecinos	1	3	5	7
Originales (sin reducir)	32,6667	29,3333	30	29,3333
2	37	33	33	29,6667
4	36	31	27,3333	28,3333
6	30,6667	25	28,6667	26,3333
8	39	31,6667	28	25
10	33,3333	27	26,3333	25,3333
12	31,6667	27	29,3333	29
14	29,3333	27,6667	27,6667	26,3333

Tabla 4.11 Error de test utilizando PCA + KNN en y KNN a los datos originales (German).

Lo que se puede observar de la tabla 4.11 es que el menor error recogido por KNN utilizando los datos facilitados por PCA es de un 25%, mientras que al aplicar KNN a los datos originales, el menor error es del 29.3333 %.

4.3.3.4 Comparación entre NN+KNN y PCA+KNN.

A continuación se va a representar con la ayuda de una tabla de resultados la diferencia entre realizar la reducción de dimensionalidad con una red de neuronas o utilizar el método PCA.

Elementos \ Nº Vecinos	1	3	5	7
2	0,3333	2	4	-0,6666
4	2,3333	-2	-1,3334	-0,3334
6	-0,6666	-6,3333	-1,3333	-3,6667
8	8,3333	3,3334	1,6667	-2
10	-3,3334	-3	-1,6667	-2
12	2	1	4	3,3333
14	-0,3334	-0,6666	0,3334	2

Tabla 4.12 Diferencia entre los errores cometidos por la red de neuronas y PCA (German).

En cada casilla se representa la diferencia que existe entre los errores recogidos por KNN al utilizar los datos obtenidos de la reducción de atributos utilizando la red de neuronas o PCA. A medida que el número que aparezca sea mayor, significará que la red de neuronas es capaz de reducir atributos de una manera más eficaz que PCA.

4.3.4 Resultados obtenidos del conjunto de datos Estate.

En este apartado se encuentran los resultados obtenidos para poder realizar el estudio de reducción de dimensionalidad utilizando la base de datos de Estate.

4.3.4.1 Error producido por una red de neuronas.

Contiene los resultados obtenidos al aplicar una red de neuronas en la que se han ido variando el número de neuronas que forman parte de la capa oculta.

Nº Neuronas	% de Error Test
1	11.3409
2	11.1529
3	11.4035
4	11.3409
5	11.1529
6	11.1529
7	11.3409

Tabla 4.13 Error de test utilizando una red de neuronas (Estate).

Se observa que el menor porcentaje de error obtenido para el conjunto de test formado por los datos de Estate es un 1.1529 % que se capta en un tripe empate al utilizar 2, 5 y 6 neuronas en la capa oculta de la red.

4.3.4.2 Error producido por NN+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos

artificial, se le facilitan los datos transformados en los cuales se ha utilizado una red de neuronas para reducir la dimensionalidad de los mismos.

Nº Vecinos / Nº Neuronas	1	3	5	7
1	19,9875	14,3484	12,5313	11,7794
2	19,7368	13,7845	11,9674	12
3	20,1754	14,2857	12,8446	11,4821
4	18,985	13,7845	12,2807	11,7794
5	19,4862	12,9699	12,4687	11,7794
6	18,7343	13,4085	12,218	11,6541
7	18,609	13,4085	12,4387	11,7794

Tabla 4.14 Error de test utilizando una red de neuronas + KNN (Estate).

Como se observa en la tabla 4.14 el menor porcentaje de error que se comete es el 11.4821% recogido en la prueba que consiste en reducir a 3 elementos utilizando la red de neuronas y utilizar para el clasificador los 7 vecinos más cercanos.

4.3.4.3 Error producido por PCA+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos girada artificialmente, los datos de los N elementos mejores que son decididos utilizando el método de PCA.

Además para poder realizar de mejor manera la comparación se añade en esta misma tabla lo resultados que se obtiene al aplicar KNN directamente a los datos originales en los que entran en juego todos los que forman el conjunto de datos girados artificialmente.

Nº Componentes \ Nº Vecinos	1	3	5	7
Originales(sin reducir)	18.5464	13.5338	12.0927	11.4035
1	17,7825	14,3484	12,594	11,6541
2	18,1704	14,2857	12,2807	11,9048
3	17,8571	14,0351	12,1554	11,7794
4	35,6516	13,8471	12,3434	11,7168
5	17,9198	13,4085	12,1554	11,7168
6	18,4837	13,5965	12,218	11,5288
7	17,8571	13,2832	12,2807	11,7168

Tabla 4.15 Error de test utilizando PCA + KNN en y KNN a los datos originales (Estate).

Lo que se puede observar de la tabla 4.15 es que el menor error recogido por KNN utilizando los datos facilitados por PCA es de un 11.5288 %, mientras que al aplicar KNN a los datos originales, el menor error es del 11.4035 %.

4.3.4.4 Comparación entre NN+KNN y PCA+KNN.

A continuación se va a representar con la ayuda de una tabla de resultados la diferencia entre realizar la reducción de dimensionalidad con una red de neuronas o utilizar el método PCA.

Elementos \ Nº Vecinos	1	3	5	7
1	-2,205	0	0,0627	-0,1253
2	-1,5664	0,5012	0,3133	-0,0952
3	-2,3183	-0,2506	-0,6892	0,2973
4	16,6666	0,0626	0,0627	-0,0626
5	-1,5664	0,4386	-0,3133	-0,0626
6	-0,2506	0,188	0	-0,1253
7	-0,7519	-0,1253	-0,158	-0,0626

Tabla 4.16 Diferencia entre los errores cometidos por la red de neuronas y PCA (Estate).

En cada casilla se representa la diferencia que existe entre los errores recogidos por KNN al utilizar los datos obtenidos de la reducción de atributos utilizando la red de neuronas o PCA. A medida que el número que aparezca sea mayor, significará que la red de neuronas es capaz de reducir atributos de una manera más eficaz que PCA.

4.3.5 Resultados obtenidos del conjunto de datos Splice.

En este apartado se encuentran los resultados obtenidos para poder realizar el estudio de reducción de dimensionalidad utilizando la base de datos Splice.

4.3.5.1 Error producido por una red de neuronas.

Contiene los resultados obtenidos al aplicar una red de neuronas en la que se han ido variando el número de neuronas que forman parte de la capa oculta.

Nº Neuronas	% de Error Test
5	29.3333
10	25
15	29.3333
20	32
25	28
30	31
35	36.3333

Tabla 4.17 Error de test utilizando una red de neuronas (Splice).

Se observa que el menor porcentaje de error obtenido para el conjunto de test formado por los datos de Splice es un 25 % que se capta en al utilizar un total de 10 neuronas en la capa oculta de la red.

4.3.5.2 Error producido por NN+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos artificial, se le facilitan los datos transformados en los cuales se ha utilizado una red de neuronas para reducir la dimensionalidad de los mismos.

Nº Vecinos / Nº Neuronas	1	3	5	7
5	29,667	29,3333	26	28,3333
10	25,3333	25	26	25,6667
15	32,667	30	29	29,3333
20	29,3333	31,3333	33	32
25	33,6667	32,6667	31	31
30	38,6667	42,3333	39,6667	37,3333
35	35,3333	33,6667	32,6667	30,667

Tabla 4.18 Error de test utilizando una red de neuronas + KNN (Splice).

Como se observa en la tabla 4.18 el menor porcentaje de error que se comete es el 25% recogido en la prueba que consiste en reducir a 10 elementos utilizando la red de neuronas y utilizar para el clasificador los 3 vecinos más cercanos.

4.3.5.3 Error producido por PCA+KNN.

A continuación se muestran los resultados expresados en porcentaje del error que comete un clasificador KNN al cual en vez de facilitarle los datos originales de la base de datos girada artificialmente, los datos de los N elementos mejores que son decididos utilizando el método de PCA.

Además para poder realizar de mejor manera la comparación se añade en esta misma tabla lo resultados que se obtiene al aplicar KNN directamente a los datos originales en los que entran en juego todos los que forman el conjunto de datos girados artificialmente.

Nº Componentes \ Nº Vecinos	1	3	5	7
Originales(sin reducir)	32	32	33.3333	33.6667
5	50,3333	47	45	46,3333
10	43	43,6667	44,3333	45,6667
15	45,3333	45,6667	43,3333	41,6667
20	42	41,6667	42,6667	41
25	42,6667	39	41	39,6667
30	35,3333	36,3333	35	35
35	29,6667	30,3333	27,6667	27,3333

Tabla 4.19 Error de test utilizando PCA + KNN en y KNN a los datos originales (Splice).

Lo que se puede observar de la tabla 4.19 es que el menor error recogido por KNN utilizando los datos facilitados por PCA es de un 27.3333 %, mientras que al aplicar KNN a los datos originales, el menor error es del 32 %.

4.3.5.4 Comparación entre NN+KNN y PCA+KNN.

A continuación se va a representar con la ayuda de una tabla de resultados la diferencia entre realizar la reducción de dimensionalidad con una red de neuronas o utilizar el método PCA.

Elementos \ Nº Vecinos	1	3	5	7
5	20,6663	17,6667	19	18
10	17,6667	18,6667	18,3333	20
15	12,6663	15,6667	14,3333	12,3334
20	12,6667	10,3334	9,6667	9
25	9	6,3333	10	8,6667
30	-3,3334	-6	-4,6667	-2,3333
35	-5,6666	-3,3334	-5	-3,3337

Tabla 4.20 Diferencia entre los errores cometidos por la red de neuronas y PCA (Splice).

En cada casilla se representa la diferencia que existe entre los errores recogidos por KNN al utilizar los datos obtenidos de la reducción de atributos utilizando la red de neuronas o PCA. A medida que el número que aparezca sea mayor, significará que la red de neuronas es capaz de reducir atributos de una manera más eficaz que PCA.

4.4 Análisis de resultados.

El propósito de este apartado es analizar los resultados obtenidos en el estudio realizado y así poder establecer, si es posible realizar una reducción de dimensionalidad a los datos originales de un dominio utilizando éstas técnicas.

Para conseguir el objetivo se van a analizar los dominios uno a uno en base a los resultados recogidos en el apartado 4.3. Se comenzará analizando los resultados particulares de cada dominio para finalmente generalizar con una visión global. La comparación se realizará de acuerdo a la figura 3.1.

4.4.1 Análisis de resultados del dominio artificial girado.

Comenzaremos comentando el único dominio artificial, es decir, aquel en que los datos utilizados no son recogidos del mundo real.

En este apartado se van a analizar los resultados obtenidos de las pruebas realizadas al dominio artificial girado. En primer lugar, se comparará la transformación y reducción de dimensionalidad realizada por PCA (PCA+KNN) y por la capa oculta de la red (NN+KNN). Si se observa la tabla 4.4 se puede ver que para todas las posibles combinaciones de número de componentes y número de vecinos, la capa oculta obtiene resultados positivos en todos los casos (26) excepto en 2, y en estos dos casos la diferencia es muy pequeña. De todas las pruebas realizadas se puede observar que en el mejor de los casos la NN+KNN trabaja un 44.5185% mejor que la reducción de dimensionalidad utilizando PCA.

A continuación se compararán los mejores resultados obtenidos por KNN después de realizar una reducción de dimensionalidad con una red de neuronas, KNN con una reducción de datos mediante PCA y KNN con todos los atributos originales.

Comenzamos con los resultados obtenidos de realizar la reducción con la capa oculta de una red de neuronas. Para este caso se observa el mejor resultado obtenido en todo el estudio, la red arroja un 1.9327% de error al clasificar los datos reducidos de dimensionalidad hasta los 7 elementos y utilizando para el algoritmo KNN los tres vecinos más cercanos. Por otro lado si se observa el resultado obtenido al aplicar KNN a los datos originales, sin realizar ninguna reducción (es decir utilizando el total de los atributos: 10), se puede observar que el mejor error que se obtiene es un 36.7877%, muy alejado del 1.9327%.

En base a este resultado realizar la reducción de dimensionalidad es muy importante, puesto que se reduce el error cometido a la hora de la clasificación y además se utilizan menos elementos para llevarla a cabo.

A continuación se realiza la reducción de dimensionalidad utilizando el método de PCA. Para comparar los resultados se aplica KNN a los datos originales (10 atributos), de esta prueba se recoge que el menor error que produce la clasificación es de un 36.7877 % y que al aplicar la reducción de dimensionalidad el menor error es de un 42.7857 %, es decir mayor que sin utilizar la reducción, por tanto PCA no reduce correctamente la dimensionalidad de este dominio y pierde información importante.

4.4.2 Análisis de resultados de Spambase.

En este apartado se comentan los resultados obtenidos del conjunto de datos Spambase.

Al igual que en el apartado anterior, comenzaremos observando y comparando la transformación y reducción de dimensionalidad realizada por la capa oculta de la red de neuronas y por PCA.

Para este dominio, se observa que para todos los casos que recoge la tabla 4.8, en sus distintas combinaciones de número de componentes y número de vecinos, la capa oculta recoge resultados positivos en comparación con PCA, siendo en el mejor de los casos hasta un 21.8310 % mejor.

A continuación analizamos los resultados obtenidos de realizar la reducción de dimensionalidad utilizando la capa oculta de la red de neuronas. Bien, si se observan los resultados de la tabla 4.6, se obtiene que el menor de los errores es un 6.9782% lo que mejora

el error de clasificación producido por KNN con los datos originales (10.5634%), pero dicho resultado se obtiene utilizando menos atributos que con los datos originales. El resultado se obtiene transformando y reduciendo todos los datos originales a 6 atributos que son los que forman las activaciones de la capa oculta. Por tanto reducir la dimensionalidad de los datos de Spambase utilizando NN+KNN ha sido un paso positivo, ya que en vez de tener que utilizar los 57 atributos originales, se podría trabajar únicamente con 6.

Para comprobar los resultados de PCA compararemos el comportamiento de los datos originales ante una clasificación con KNN con los datos facilitados tras aplicarles el método de PCA. Con los datos originales el mejor resultado que se obtiene con KNN es un 10.5634% de error cuando se utilizan los tres vecinos más cercanos a la hora de clasificar. Tabla 4.7. Al utilizar la reducción de atributos con PCA, el mejor resultado se obtiene en la prueba en la que entran en juego más componentes (26), y el porcentaje de error aumenta hasta el 11.9078%. Por tanto realizar una reducción de dimensionalidad con PCA no mejora el error que se comete al no reducir la dimensionalidad de los datos y utilizar KNN, y es mucho mayor si se compara con la reducción de dimensionalidad realizada con la red de neuronas.

Por tanto para este dominio, al igual que en el anterior, es mucho más eficaz utilizar la transformación y reducción de dimensionalidad mediante una red de neuronas artificiales frente a realizar la reducción de dimensionalidad con PCA.

4.4.3 Análisis de resultados de German.

A continuación se van a analizar los resultados obtenidos de realizar las pruebas establecidas en la metodología a la base de datos German.

Para comenzar se comparan los resultados obtenidos de realizar la transformación y reducción de dimensionalidad mediante una red de neuronas y mediante PCA. Los resultados están recogidos en la tabla 4.12, y se observa que de los 28 resultados, 13 son números positivos y 15 son negativos, por tanto en la mayoría de los casos realizar la reducción de dimensionalidad mediante una red de neuronas no recoge mejores resultados que realizarla con PCA. Sin embargo, en aquellos casos en los que PCA funciona mejor que la capa oculta, las diferencias no son grandes (en 14 de los 15 casos, son menores que 3.66%).

Seguidamente se analizan los resultados obtenidos al realizar una clasificación utilizando KNN en la que entran en juego los atributos reducidos de dimensionalidad gracias a

la capa oculta de la red de neuronas. El mejor resultado obtenido es un 24.3333% que se recoge al reducir a 14 atributos los datos originales (24 atributos). Si se compara con el mejor dato recogido mediante KNN con los datos originales se puede afirmar que la reducción de atributos mejora el resultado puesto que el error que comete es de 29.3333%.

La siguiente prueba de la que se recogen resultados para el dominio German es en la que se reduce la dimensionad de los datos utilizando PCA y a continuación se realiza una clasificación utilizando KNN. Al realizar la clasificación de los datos sin reducirlos, se observa que el menor porcentaje de error que se produce es un 29.3333 % cuando se utilizan 3 o 7 vecinos. Sin embargo al realizar la reducción de atributos mediante el método PCA y quedarnos únicamente con los 8 mejores atributos y realizar una clasificación con KNN en la que entran en juego los 7 vecinos más cercanos, se produce un error del 25%, lo que mejora el resultado obtenido al utilizar KNN con los datos originales.

Resumiendo el análisis anterior, también el mejor de los resultados de NN+KNN es mejor que de los datos originales y NN+PCA.

4.4.4 Análisis de resultados de Estate.

El siguiente grupo de resultados que se van a analizar son aquellos que se obtiene del conjunto de datos Estate.

Comenzamos comparando los errores cometidos por NN+KNN y por PCA+KNN. Se observa que ambas reducciones trabajan de forma muy similar, de los 28 resultados obtenidos 11 son positivos y 17 negativos, pero en aquellos casos en los que NN+KNN funciona peor que PCA+KNN, las diferencias son muy pequeñas (inferiores al 2.3%).

A continuación se analiza los resultados obtenidos al reducir la dimensionalidad de los datos utilizando la capa oculta de una red de neuronas y seguidamente aplicando un clasificador KNN. En esta prueba el menor error que se recoge es un 11.4821 %, que es solo ligeramente mayor al que se produce utilizando únicamente KNN para clasificar (11.4035%). Sin embargo éste error es producido con únicamente 3 elementos y usando los 7 vecinos más cercanos para el clasificador KNN, frente a los 12 atributos usados originalmente. Por tanto aunque el error es ligeramente superior, la ventaja de utilizar únicamente 3 atributos hace que se pueda afirmar que la reducción de dimensionalidad cumple su función. Si se hace la reducción de atributos con PCA, el mejor resultado es un 11.5288% que se obtiene con 6

componentes y 7 vecinos. Este resultado es peor que KNN con los datos originales y que NN+KNN, con lo que aunque en este dominio todas las técnicas obtienen resultados muy similares, es NN+KNN quien consigue reducir en mayor medida el número de atributos.

Al igual que en los anteriores dominios también se realiza una reducción de dimensionalidad utilizando PCA y aplicando KNN a los resultados obtenidos. También se aplica KNN a los datos originales (12 atributos), para dicha prueba se recoge un 11.4035 % de error al utilizar los 7 vecinos más cercanos. Si observamos los resultados obtenidos para la clasificación con los datos reducidos mediante PCA, observamos que el mejor de los errores es un 11.5288 % que se obtiene utilizando 6 componentes y 7 vecinos en la clasificación. Por tanto se mantiene cercano el valor del error cometido, pero sin embargo al utilizar redes de neuronas el error es menor y son necesarios menos elementos para obtener el resultado.

4.4.5 Análisis de resultados de Splice.

Por último se va a analizar los resultados que se obtienen de realizar las pruebas al conjunto de datos de Splice.

Si se observa el comparativo entre los errores cometidos por NN+KNN y por PCA+KNN, en la tabla 4.20, se observa que en la mayoría de los casos (de 28 pruebas, 20), NN+KNN se comporta mejor a la hora de clasificar en comparación con PCA+KNN, siendo en el mejor de los casos hasta un 20.6663 % mejor. También se puede observar que NN+KNN se comporta mejor, cuando se reducen los atributos que se utilizan para la clasificación, es decir, que a menor cantidad de atributos, aplicar NN+KNN consigue mejores resultados que PCA+KNN.

A continuación se analizan los resultados recogidos de realizar la reducción de dimensionalidad mediante la capa oculta de una red de neuronas, y aplicando a los resultados obtenidos un clasificador KNN. Observando los resultados de la tabla 4.18 se observa que el método de reducción de dimensionalidad ha conseguido igualar el error de clasificación a un 25% utilizando sólo 10 elementos y entrando en juego los 3 vecinos más cercanos para KNN.

Ahora observamos si el método de reducción de dimensionalidad mediante PCA es capaz de trabajar de mejor manera. Aplicamos KNN a los datos originales, los cuales están formados por 60 atributos y observamos que recogen en el mejor de los casos un 32 % de error al utilizar el vecino más cercano o los 3 vecinos más cercanos. Después de la reducción de dimensionalidad mediante PCA el error disminuye al 27.333 % utilizando los 35 mejores

componentes y que en KNN entren en juego los 7 mejores vecinos. Por tanto de nuevo NN+KNN es capaz de reducir la dimensionalidad de los datos de mejor manera que PCA+KNN.

Por tanto, para este dominio, reducir la dimensionalidad mediante la capa oculta de una red de neuronas artificiales es una técnica válida que recoge mejores resultados que realizar la reducción de dimensionalidad con PCA o utilizar KNN con el conjunto de datos original.

4.4.6 Resumen del análisis de datos.

A continuación se muestra un resumen de resultados para que estén todos agrupados y se puedan observar de una manera más sencilla.

	KNN (Atributos originales)	PCA+KNN (Componentes)	NN+KNN (Nº atributos)	Veces que la capa oculta mejora a PCA
Dominio Artificial	36,8% (10)	42,8% (7)	1,9% (7)	26 de 28
Spambase	10,6% (57)	11,9% (26)	7% (6)	28 de 28
German	29,3% (20)	29,3% (8)	24,3% (14)	13 de 28
Estate	11,4% (12)	11,5% (6)	11,5% (3)	11 de 28
Splice	32% (60)	27,3% (35)	25% (10)	20 de 28

Tabla 4.21 Recopilación de los mejores resultados.

En la tabla 4.21 se muestran los menores porcentajes de error cometidos por KNN para cada dominio, dando como resultado 3 columnas formadas por los errores de KNN aplicado al conjunto de datos original, KNN aplicado a los atributos reducidos mediante PCA y KNN aplicado al dominio transformado y reducido mediante la capa oculta de una red de neuronas. Entre paréntesis aparecen el número de atributos. Finalmente, se añade una columna más en la que se muestra el número de ocasiones de las totales en las que NN+KNN se comporta mejor que PCA+KNN reduciendo los atributos. Todos estos resultados están plasmados con más detalle desde la tabla 4.1 hasta la tabla 4.20.

En la primera columna de la tabla 4.21 llamada KNN (Atributos originales) se muestran los menores porcentajes de error cometidos al aplicar una clasificación mediante KNN a los

atributos originales del dominio. Además entre paréntesis se muestra el número total de atributos que forman el dominio utilizado.

En la segunda columna se muestran los menores porcentajes de error cometido al realizar una clasificación con KNN utilizando para ello los datos reducidos de dimensionalidad mediante PCA. Entre paréntesis se muestra el número de componentes con los que se recoge cada error.

La tercera columna contiene el menor porcentaje de error cometido por KNN al clasificar los atributos después de realizar una transformación y reducción de dimensionalidad mediante la capa oculta de una red de neuronas. Entre paréntesis se muestra el número de neuronas utilizadas para la capa oculta, que se traduce en el número de atributos finales que se utilizan.

Por último la cuarta columna indica el número de pruebas en las que la reducción de dimensionalidad realizada con una red de neuronas (NN+KNN) mejora al resultado obtenido con una reducción de dimensionalidad utilizando PCA (PCA+KNN).

De las tablas se puede concluir que realizar la reducción de dimensionalidad utilizando NN+KNN es para todos de los casos una opción muy válida, puesto que se recogen menores porcentajes de error en todas las pruebas, y en aquellas en las que no, el error se encuentra muy cercano y por tanto puede merecer la pena realizar la reducción puesto que el número de componentes es mucho menor.

Según recogen los resultados, la reducción de dimensionalidad utilizando PCA, no mejora los resultados obtenidos al realizar una clasificación utilizando los datos originales para la mayoría de los casos. Únicamente se produce una excepción con el dominio Splice.

Con respecto a la última columna, podemos distinguir dos casos. Para el caso del dominio artificial Spambase y Splice, NN+KNN obtiene mejores resultados que PCA+KNN, casi siempre, esto es, independientemente del número de componentes y número de vecinos. Para German y Estate esto no es así, pero como ya se comentó anteriormente, para aquellos casos en los que NN+KNN funciona peor que PCA+KNN, las diferencias no son grandes. Y en cualquier caso, como ya se ha comentado en los párrafos anteriores, los mejores resultados de NN+KNN son comparables o ligeramente mejores también en estos casos.

Por tanto como conclusión final: para realizar una reducción de dimensionalidad, se puede recomendar el método que transforma y reduce los atributos de un dominio utilizando la capa oculta de una red de neuronas.

Capítulo 5: Planificación y presupuesto

5 Planificación y presupuesto.

En este apartado se va a exponer la planificación utilizada para la realización de éste trabajo de fin de grado así como el presupuesto en el que se estiman los gastos necesarios para poder llevarlo a cabo.

5.1 Planificación.

El trabajo de fin de grado se ha realizado a lo largo de los dos últimos cuatrimestres del último curso del Grado en Ingeniería Informática. Se le asigna un peso total de 12 ECTS, cada ECTS se asigna unas 25 horas de trabajo, por tanto la duración del proyecto debe de ser, al menos de unas 300 horas.

El estudio realizado en este trabajo se ha tomado como si fuera un proyecto real, en el que han entrado en juego diferentes fases que pasan a ser representadas mediante el siguiente diagrama de Gantt.

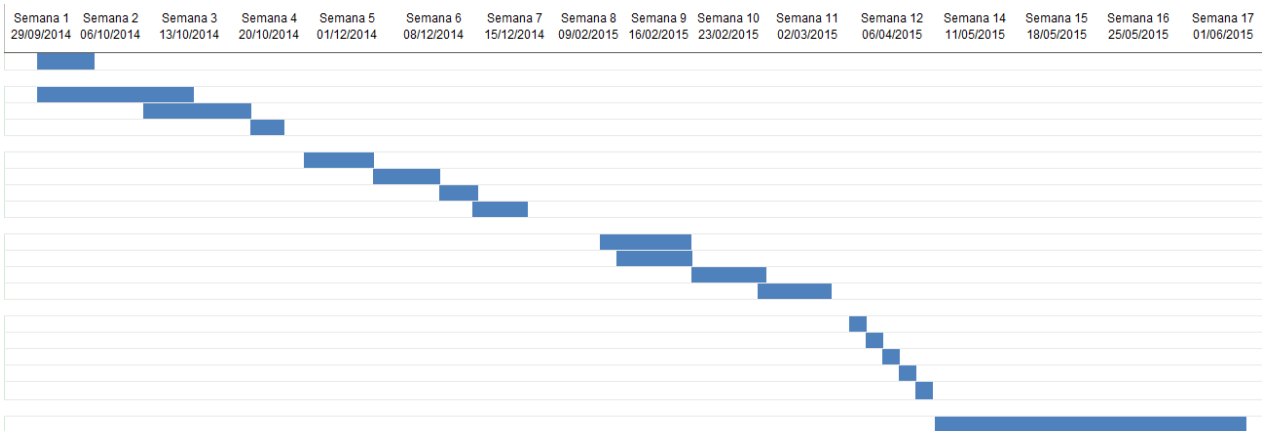

Figura 5.1 Diagrama de Gantt.

Seguidamente se muestra un detalle de todas las tareas representadas en el diagrama de Gantt de la figura 5.1.

Id	Tareas	Comienzo	Final	Cal. Días
0	[Proposición de proyecto]	01/10/2014	06/10/2014	6
1	**[Análisis]**			24
1.1	[Estudio Estado del arte]	01/10/2014	15/10/2014	15
1.2	[Planteamiento del problema]	16/10/2014	22/10/2014	7
1.3	[Toma de requisitos]	23/10/2014	24/10/2014	2
2	**[Diseño]**			18
2.1	[Obtener activaciones capa oculta]	01/12/2014	05/12/2014	5
2.2	[Algoritmo KNN]	06/12/2014	10/12/2014	5
2.3	[Algoritmo PCA]	11/12/2014	13/12/2014	3
2.4	[Aplicar KNN a las diferentes técnicas]	13/12/2014	17/12/2014	5
3	**[Implementación]**			32
3.1	[Obtener activaciones]	12/02/2015	21/02/2015	10
3.2	[Aplicar KNN a las activaciones]	13/02/2015	22/02/2015	10
3.3	[Aplicar KNN a los datos originales]	14/02/2015	18/02/2015	5
3.4	[Aplicar PCA y KNN]	15/02/2015	21/02/2015	7
4	**[Obtención de resultados]**			5
4.1	[Resultados de Spambase]	06/04/2015	06/04/2015	1
4.2	[Resultados de German]	07/04/2015	07/04/2015	1
4.3	[Resultados del dominio girado]	08/04/2015	08/04/2015	1
4.4	[Resultados de Estate]	09/04/2015	09/04/2015	1
4.5	[Resultados de Splice]	10/04/2015	10/04/2015	1
5	**[Documentación]**			25
5.1	[Redacción de documentación]	11/05/2015	04/06/2015	25

Tabla 5.1 Detalle del diagrama de Gantt.

El Id es el identificador de la tarea, para que ésta sea representada de manera única, a continuación se expone el nombre de la tarea, repartidas en diferentes bloques. A cada tarea se asigna una fecha de inicio y una fecha de fin, así como los días que debe durar su ejecución. Además en la información sobre los días de duración para los bloques se indica el número total de días utilizados para la realización de cada bloque.

5.2 Presupuesto.

En el apartado de presupuesto se calculará, en base a la planificación del apartado anterior, el gasto que es necesario para poder llevar a cabo el proyecto. El presupuesto incluye los gastos de personal, hardware, software y posibles gastos añadidos.

5.2.2 Gastos de hardware.

En este apartado, se expone los gastos que vienen producidos por la utilización del hardware necesario para para el desarrollo y creación del sistema con el que se recogerán los resultados necesarios para el estudio.

El único hardware necesario es un ordenador de sobremesa suficientemente potente como para realizar los cálculos de una manera solvente.

Para calcular el gasto generado por la utilización del hardware se utiliza la siguiente fórmula en la que entra en juego las semanas dedicadas, el periodo de depreciación, el coste y el porcentaje de uso:

$$GastoDeHardware = \frac{SemanasUtilizado}{PeriodoDeDepreciación}\ x\ Coste\ x\ PorcentajeUso$$

Ecuación 5.1 Gasto de hardware.

El equipo tiene un coste de 750€, se ha utilizado durante un total de 22 semanas, el periodo de depreciación es de 52 semanas, su porcentaje de uso es de un 70 %. Por tanto el coste total del hardware es:

$$GastoDeHardware = \frac{22}{52}\ x\ 750\ x\ 0.7 = 222.12€$$

Ecuación 5.2 Gasto total de hardware en euros.

5.2.3 Gastos de software.

A continuación se muestran los gastos producidos por el uso de software.

Mediante la plataforma MSDNAA que pone a disposición del alumno la Universidad Carlos III de Madrid, mucho del software no ha tenido que ser adquirido y ha producido que la reducción del presupuesto en este apartado haya sido considerable. También ha ayudado que el resto de software que no se puede adquirir a través de MSDNAA está instalado en las aulas informáticas de la universidad.

El gasto ha sido reducido considerablemente porque todo el software utilizado es privado y por tanto tiene un alto coste. El software utilizado ha sido: Windows 7 x 64 Proffesional, Microsoft Office y Matlab. En la siguiente tabla se muestra el total de los costes.

Software	Gasto
Windows 7 x64 Proffesional	0 €
Microsoft Office	0 €
Matlab	0 €
Total	0 €

Tabla 5.2 Gasto total de software en euros.

5.2.4 Gastos de personal.

Otro gasto a tener en cuenta es el gasto de personal, que está formado por el sueldo que se debe pagar al desarrollador durante la duración del proyecto.

Se calcula un salario medio de unos 24.000 € al año, teniendo en cuenta que la duración del proyecto ha sido en total de 22 semanas en la que cada jornada laboral cuenta con 20 horas por semana, el gasto de personal al que hay que hacer frente es de:

$$Gastos\ de\ personal = \ 24000 * \frac{22}{52} x \frac{20}{40} = 5076.62\ €$$

Ecuación 5.3 Gasto total de personal en euros.

5.2.4 Resto de gastos.

A parte de los gastos anteriormente mencionados, hay que añadir otro tipo de gastos indirectos del proyecto como pueden ser el gasto de material, márgenes de error y beneficio.

Los gastos en material se han fijado en unos 20 euros en los que entran el gasto de impresiones, folios, grapas, tinta, desgaste de la impresora… También se deben tener en cuenta los gastos que conlleva lo anteriormente detallado como puede ser la luz, la conexión a internet, alquiler de oficina… Por todo ello se decide aplicar un 15 % al total de los gastos. También se ha de asegurar que el proyecto sea viable y para ello se cuenta con 10% de margen de error. Por último como margen de beneficio se establecerá un 25 %.

5.2.5 Total gastos.

Una vez que se han declarado todos los gastos que produce llevar a cabo este proyecto, se reúne todo en la siguiente tabla que contiene el gasto total del proyecto:

Concepto	Valor
Gastos directos	***5318.74***
Hardware	222.12 €
Software	0 €
Personal	5076.62 €
Gastos varios	20 €
Gastos indirectos (15%)	***797.81 €***
Total	***6116.55 €***
Riesgo (10%)	611.65 €
Beneficios (25%)	1529.14 €
Total Sin I.V.A	***8257.34 €***
I.V.A (21%)	1734.04 €
Total presupuesto	**9991.38 €**

Tabla 5.3 Presupuesto total.

Como se puede observar después de realizar los cálculos se monta una cantidad total de 9991.38 Euros como coste final del trabajo.

Capítulo 6: Marco Regulador

6. Marco regulador.

Este trabajo de fin de grado ha sido realizado tomando en cuenta el ámbito legal que a éste pudiera afectar.

El uso de Matlab ha sido bajo la licencia que la UC3M posee de Matlab, la cual permite el uso del software en el hardware de la universidad para llevar a cabo los trabajos del curso y realizar investigaciones académicas en instituciones que otorgan títulos. No está disponible para uso comercial, gobierno u otros usos para organizaciones diferentes.

Gracias a las licencias que la UC3M posee de Windows 7 x64 Proffesional y Microsoft Office, se puede utilizar dicho software en el hardware de la universidad para llevar a cabo los trabajos necesarios en el curso, así como realizar investigaciones académicas. No está disponible para uso comercial, gobierno u otros usos para organizaciones diferentes.

Las bases de datos utilizadas para la obtención de resultados son propiedad de KEEL. Página que permite descargar y usar todas las bases de datos que se encuentran en el repositorio para analizar el comportamiento de los métodos de aprendizaje. La condición que impone la página es que siempre se cite en las referencias del trabajo presentado con un formato establecido.

Capítulo 7: Conclusiones y futuros trabajos

7. Conclusiones.

En este apartado, se exponen las conclusiones obtenidas de la realización de este trabajo de fin de grado.

El objetivo principal del trabajo es realizar un estudio de reducción de dimensionalidad para problemas supervisados. Pero para poder realizar dicho estudio, se necesita de un programa que sea capaz de obtener los resultados necesarios para realizar dicho estudio.

Para comenzar, se exponen que la aplicación necesaria para la obtención de resultados necesarios para realizar el estudio fijado para este proyecto ha sido realizada de manera correcta y realiza todos los objetivos fijados:

- Es capaz de realizar una transformación y reducción de dimensionalidad utilizando la capa oculta de una red de neuronas, teniendo que extraer los atributos de la capa oculta de la red.
- Es capaz de realizar una reducción de dimensionalidad utilizando el método de PCA.
- Es capaz de aplicar KNN a los datos originales, a los datos transformados y reducidos mediante la capa oculta de una red de neuronas y a los datos reducidos utilizando el método PCA.

Por lo tanto, todos los objetivos fijados para la aplicación son realizados de manera satisfactoria.

El siguiente objetivo es realizar un estudio en el que se comprobase si es posible que al realizar una reducción de dimensionalidad utilizando la capa oculta de una red de neuronas, los atributos reducidos fuesen capaces de generalizar el conocimiento de mejor manera que los atributos reducidos mediante uno de los métodos de reducción más utilizados como es PCA.

En el apartado 4: Estudio realizado, se muestran todos los resultados del estudio. Gracias a los resultados que se plasman en el estudio, se puede concluir que realizar una reducción de dimensionalidad mediante la capa oculta de una red de neuronas, arroja mejores resultados que realizar la reducción de dimensionalidad mediante el método PCA. Si nos centramos en los dominios utilizados, se observa que para el dominio artificial tanto como para Spambase y Splice, NN+KNN obtiene mejores resultados que PCA+KNN en la mayoría de las ocasiones, esto es, independientemente del número de componentes y número de vecinos que entren en juego. Para los otros dos dominios German y Estate, esto no es así, pero para aquellos casos en los que NN+KNN funciona peor que PCA+KNN, las diferencias no son grandes. De cualquier manera los mejores resultados de NN+KNN son comparables o ligeramente mejores también para estos casos.

Por lo tanto, por todo lo anterior, todos los objetivos fijados han sido realizados de manera correcta y además se ha encontrado una nueva técnica de reducción (utilizar la capa oculta de una red de neuronas artificiales) que funciona mejor reduciendo la dimensionalidad de los atributos que una de las que más se utilizan en la actualidad: PCA.

7.1 Conclusiones personales.

Como conclusión personal, gracias a la realización de este trabajo de fin de grado, he podido aplicar los diferentes conocimientos que he ido adquiriendo a lo largo del grado.

Además realizar un trabajo de investigación es emocionante, ya que trabajas sobre algo que nadie antes ha planteado y por tanto llevarlo a cabo y que termine de manera satisfactoria es algo muy gratificante.

7.2 Trabajos futuros.

Ya que este trabajo de grado es un estudio, y por tanto un trabajo de investigación, los trabajos futuros son muy amplios.

Se puede experimentar con muchos más dominios para seguir observando el comportamiento que presenta, utilizar nuevos dominios reales e incluso se podría

experimentar con dominios reales en los que se añadan atributos aleatorios, de manera similar a lo que se realiza con el dominio artificial girado de este trabajo.

También se puede comprobar el comportamiento de la reducción de dimensionalidad mediante la capa oculta de una red de neuronas frente a otros métodos de reducción de dimensionalidad diferentes al utilizado en este trabajo (PCA).

Otro trabajo futuro será realizar estudios en los que se compruebe cómo se comporta esta transformación y reducción de atributos con otros clasificadores que no sean KNN.

Capítulo 8: Bibliografía

8. Bibliografía.

[1] **Isasi Viñuela, P, Galván León, I**. (2004) Redes de neuronas artificiales. Un enfoque práctico. Pearson Educación S.A.

[2] **KEEL-dataset citation paper:** J. Alcalá-Fdez, A. Fernandez, J. Luengo, J. Derrac, S. García, L. Sánchez, F. Herrera. KEEL Data-Mining Software Tool: Data Set Repository, Integration of Algorithms and Experimental Analysis Framework. Journal of Multiple-Valued Logic and Soft Computing 17:2-3 (2011) 255-287

[3] **Mur, R. A.** (08/02/2013). Aprendizaje Automático para el Análisis de Datos. Obtenido el 10/11/2014, desde el sitio Web de OCW - UC3M: http://ocw.uc3m.es/ingenieria-informatica/aprendizaje-automatico-para-el-analisis-de-datos-2013.

[4] **Mur, R. A.** (29/10/2007). Material de clase. Obtenido el 10/11/2014, desde el sitio Web de OCW - UC3M: http://ocw.uc3m.es/ingenieria-informatica/aprendizaje-automatico-para-el-analisis-de-datos-2013/material-de-clase.

[5] **Valls, J. M.** (2007). Material de clase. Obtenido el 10/11/2014, desde el sitio Web de EVANNAI - UC3M: http://eva.evannai.inf.uc3m.es/et/docencia/rn-inf/documentacion/Tema3-MLP.pdf

[6] **Grané, A**. (s.d.). Ficheros de docencia. Obtenido el 15/11/2014, desde el sitio Web HALWEB- UC3M: http://halweb.uc3m.es/esp/Personal/personas/agrane/ficheros_docencia/MULTIVARIANT/slides_comp_reducido.pdf

[7] **Marín, J. M.** (s.d.). Ficheros de docencia. Obtenido el 15/11/2014, desde el sitio Web HALWEB- UC3M: http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/DM/tema3dm.pdf

[8] **Gestal, M.** (s.d.). Introducción a las Redes de Neuronas. Obtenido el 15/12/2014, desde el sitio Web SABIA–TIC-UDC: http://sabia.tic.udc.es/mgestal/cv/RNAtutorial/TutorialRNA.pdf

[9] **Altman, N. S.** (1992). "An introduction to kernel and nearest-neighbor nonparametric regression". *The American Statistician* **46** (3): 175–185.

[10] **Redes de Neuronas Artificiales** (s.d.) Recuperado del sitio web de la UC3M: http://www.lab.inf.uc3m.es/~a0080630/redes-de-neuronas/

[11] **Smith L. I.** (26/02/2002) A tutorial on Principal Components Analysis. Obtenido desde el sitio web CS-OTAGO-AC: http://www.cs.otago.ac.nz/cosc453/student_tutorials/principal_components.pdf

Capítulo 9: Abstract

9. Abstract.

9.1 Introduction and targets.

9.1.1 Introduction

In what concerns the computer science, one of the main challenges has always been building machines that have learning capacity, able to improve automatically based on their experience, because if that were possible, a world of development and new applications would be open. On the other hand, it also would help to understand the limits of human learning.

Human beings are capable of learning and reasoning through the network of neurons that we have in the brain. Artificial neural networks are inspired by the way the animals nervous system works. It is a learning paradigm and automatic processing by which a series of entries transform and generate the outputs. This technique is used extensively in the different fields of engineering and science to solve complex problems in which more traditional techniques, such as linear or polynomial regression, do not work as well as it is expected of them.

Neural networks create a non-explicit model that relates the set of inputs in order to generate one or more output values. This requires a set of input variables observations, that generate training patterns that allow the network to be able to learn and predict an output when new observations are introduced.

Therefore it is important to differentiate the two basic phases neural networks comprise: training and testing. A set of observations are provided to the network to perform the calculations and generalize a model of data. Once the net is trained, we start the test phase in which new data (not used for training phase) check if the network has managed to generalize correctly those new observations.

It is easy to see that training phase is the most important one, and it is precisely for this reason that there are different algorithms used in it. The user needs to specify which method will be used to collect the best results.

In this EOG we are going to use backpropagation neural networks (multilayer perceptron), nonparametric techniques such as KNN and dimensionality reduction techniques like PCA.

9.1.2 Targets .

The target of this EOG is to use different dimensionality reduction techniques over a set of data, and observe if they don't lose important information in the parameters reduction and if they can pick good results by classifying new instances using a lower number of inputs. Achieve this data reduction would be a great advantage, because in using machine learning techniques the number of elements employed is directly proportional to the time required for problem resolution.

Therefore, this project is a research one in which it is not known if the results to be obtained will be favourable or not. However, if the results are positive, a door opens to further investigations or to start using these reduction techniques when needed to solve problems with a large number of attributes.

We need to emphasize that nowadays Big Data is in a rising time, and when it comes to solving problems, to reduce the data size without losing information is a very important advantage.

The different targets that we must achieve to carry out this study are:

- To conduct a dimensionality reduction using a neural network.
- To extract the attributes from the hidden layer of the neural network.
- To apply KNN to that collected attributes.
- To conduct a dimensionality reduction using PCA method, in order to apply KNN to the new data.
- To apply KNN to the original data.
- To use a neural network with the original data.
- To use the data collected in the previous section for a behaviour study.

9.2 Results.

9.2.1 Methodology.

When conducting a research study is important to determine the methodology to be used before get the data results, because all that tests will be compared and then we will be able to know the general behaviour of the system.

In this case, different techniques for dimensionality reduction will be used, in which the important variables can make the system able to classify in a better or worse way.

It is important to remember the general idea of the EOG: to realize a research work in which we will found if it is possible to reduce the attributes of a problem using different methods, and to check the system's operation once these parameters are offside.

Therefore it has been decided that for testing the attribute reduction will be from 50 % onwards, that is, the test to be performed will use 50% or less of the domain's original data. With that aggressive performance, if the system collects good results, performing the attribute reduction would be justified, because the computation weight and the execution time would be reduced considerably.

One decided that we are going to use up to 50 % of the original data set, we should distribute the test that have to be performed. For this we have decided to use seven general architectures, in which the important parameters will vary in certain cases to check their behaviour.

In the dimensionality reduction, seven architectures will be used, and in each of them the main variable will vary in a progressive and steadily way.

Once the dimensionality reduction has been performed, a classifier will be used to know how correctly and productive has been the reduction. We will use a KNN classifier. The methodology used for this step will be to create four architectures, in which will come into play 1, 3, 5, and 7 neighbours.

9.2.2 Tests.

Here are the results of the study, in which come into play both the dataset and the methodology.

9.2.2.1 Results of the Spambase database.

In this section are the results obtained for the study of dimensionality reduction using the Spambase data set.

Neural network error .

It contains the results obtained by applying a neural network in which we have been varying the number of the hidden layer neurons.

Neurons	Test Error %
2	8,0026
6	8,2546
10	7,6184
14	7,9385
18	8,8348
22	7,6184
26	8,3227

Table 9.1 Test error by using a neural network (Spambase).

It is noted that the lower percentage of error obtained for the set of test data is a 7.6184, captured twice, when the number of hidden neurons are 10 or 22.

Error occurred when reducing dimensionality using a neural network.

The results are expressed as a percentage of error that makes a KNN classifier, to which rather than be provided with the original database, it is provided with the transformed data which has been reduced using a neural network.

Neighbours / Neurons	1	3	5	7
2	10,5634	8,7068	8,3867	8,5787
6	8,3227	6,9782	7,0423	6,9782
10	8,2586	7,3624	7,0423	7,6825
14	8,3227	8,0026	7,8105	8,0026
18	9,1549	8,0666	8,3867	7,8754
22	8,3867	8,3227	8,1946	8,2586
26	9,0909	7,2343	7,7465	7,8745

Table 9.2 Test error using a neural network + KNN (Spambase).

As it is shown in table 9.2, the lowest percentage of error made is 6.9782 %, collected in the test that reduces to 6 elements using a neural network, and using 3 or 7 neighbours in KNN classifier.

Error occurred when reducing dimensionality using PCA.

In this section are the results, expressed as a percentage of error, which makes a KNN classifier when using the N best elements of the database decided by PCA method.

In order to make a better comparison with the results obtained, the section contains the results obtained by applying KNN directly to the original data.

Neighbours / Components	1	3	5	7
Original data	10,9475	10,5634	11,0755	11,6517
2	31,306	29,8976	30,2177	30,2177
6	24,1997	23,1754	23,6876	23,7516
10	18,5019	18,1818	18,758	17,9257
14	18,3739	16,6453	17,3496	17,8617
18	15,3009	16,0051	16,5813	16,9654
22	13,6364	13,5723	15,1088	15,4289
26	11,9078	12,2919	12,8681	13,8284

Table 9.3 Test error using PCA+KNN and Test error using KNN with original data (Spambase).

What can be seen from table 9.3 is that the slightest error using PCA + KNN is an 11.9078 %, while KNN using original data catch a 10.5634%.

Neural network and PCA error comparison.

Then it will be represented, with the help of a table of results, the difference between performing dimensionality reduction using a neural network or using the PCA method.

Neighbours / Elements	1	3	5	7
2	20,7426	21,1908	21,831	21,639
6	15,877	16,1972	16,6453	16,7734
10	10,2433	10,8194	11,7157	10,2432
14	10,0512	8,6427	9,5391	9,8591
18	6,146	7,9385	8,1946	9,09
22	5,2497	5,2496	6,9142	7,1703
26	2,8169	5,0576	5,1216	5,9539

Table 9.4 Error difference committed by neural network and PCA (Spambase).

In each square the number represents the difference between the errors KNN collected by using data obtained from the reduction of attributes using the PCA method or neural network. The higher the number is, the more effective the neural network attribute reduction will be, when comparing to PCA.

9.2.2.2 Results of the German database.

In this section are the results obtained for the study of dimensionality reduction using the German data set.

Neural network error .

It contains the results obtained by applying a neural network in which we have been varying the number of the hidden layer neurons.

Neurons	Test Error %
2	31,3333
4	29,6667
6	28,3333
8	27
10	29,6667
12	32,3333
14	26,6667

Table 9.5 Test error by using a neural network (German).

It is noted that the lower percentage of error obtained for the set of test data is a 26.6667 %, captured when the number of hidden neurons are 14.

Error occurred when reducing dimensionality using a neural network .

The results are expressed as a percentage of error that makes a KNN classifier, to which rather than be provided with the original database, it is provided with the transformed data which has been reduced using a neural network.

Neurons \ Neighbours	1	3	5	7
2	36,6667	31	29	30,3333
4	33,6667	33	28,6667	28,6667
6	31,3333	31,3333	30	30
8	30,6667	28,3333	26,3333	27
10	36,6667	30	28	27,3333
12	29,6667	26	25,3333	25,6667
14	29,6667	28,3333	27,3333	24,3333

Table 9.6 Test error using a neural network + KNN (German).

At it is shown in table 9.6, the lowest percentage of error made is 24.3333 %, collected in the test that reduces to 14 elements using a neural network, and using 7 neighbours in KNN classifier.

Error occurred when reducing dimensionality using PCA.

In this section are the results, expressed as a percentage of error, which makes a KNN classifier when using the N best elements of the database decided by PCA method.

In order to make a better comparison with the results obtained, the section contains the results obtained by applying KNN directly to the original data.

Components \ Neighbours	1	3	5	7
Original data	32,6667	29,3333	30	29,3333
2	37	33	33	29,6667
4	36	31	27,3333	28,3333
6	30,6667	25	28,6667	26,3333
8	39	31,6667	28	25
10	33,3333	27	26,3333	25,3333
12	31,6667	27	29,3333	29
14	29,3333	27,6667	27,6667	26,3333

Table 9.7 Test error using PCA+KNN and Test error using KNN with original data (German).

What can be seen from table 9.7 is that the slightest error using PCA + KNN is a 25 % while KNN using original data catch a 29.3333%.

Neural network and PCA error comparison.

Then it will be represented, with the help of a table of results, the difference between performing dimensionality reduction using a neural network or using the PCA method.

Neighbours / Elements	1	3	5	7
2	0,3333	2	4	-0,6666
4	2,3333	-2	-1,3334	-0,3334
6	-0,6666	-6,3333	-1,3333	-3,6667
8	8,3333	3,3334	1,6667	-2
10	-3,3334	-3	-1,6667	-2
12	2	1	4	3,3333
14	-0,3334	-0,6666	0,3334	2

Table 9.8 Error difference committed by neural network and PCA (German).

In each square the number represents the difference between the errors KNN collected by using data obtained from the reduction of attributes using the PCA method or neural network. The higher the number is, the more effective the neural network attribute reduction will be, when comparing to PCA.

9.2.2.3 Results of the artificial turned database.

In this section are the results obtained for the study of dimensionality reduction using the artificial turned data set.

Neural network error.

It contains the results obtained by applying a neural network in which we have been varying the number of the hidden layer neurons.

Neurons	Test error %
1	39.4535
2	35.8880
3	36.3545
4	38.8204
5	21.3595
6	15.1283
7	7.6974

Table 9.9 Test error by using a neural network (Artificial).

It is noted that the lower percentage of error obtained for the set of test data is a 7.6975 %, captured when the number of hidden neurons are 7.

Error occurred when reducing dimensionality using a neural network.

The results are expressed as a percentage of error that makes a KNN classifier, to which rather than be provided with the original database, it is provided with the transformed data which has been reduced using a neural network.

Neurons \ Neighbours	1	3	5	7
1	45.3182	44.9184	36.7877	43.9853
2	39.2203	37.9540	37.1876	36.5545
3	37.1543	36.8877	35.5215	34.9550
4	39.2536	37.1876	35.7881	35.2216
5	33.8554	35.8581	35.6548	35.0883
6	4.9317	4.4319	4.4985	4.2652
7	2.0993	1.9327	2.0660	2.0660

Table 9.10 Test error using a neural network + KNN (Artificial).

At it is shown in table 9.10 the lowest percentage of error made is 1.9327 %, collected in the test that reduces to 7 elements using a neural network, and using 3 neighbours in KNN classifier.

Error occurred when reducing dimensionality using PCA.

In this section are the results, expressed as a percentage of error, which makes a KNN classifier when using the N best elements of the database decided by PCA method.

In order to make a better comparison with the results obtained, the section contains the results obtained by applying KNN directly to the original data.

Components \ Neighbours	1	3	5	7
Original data	38.3206	36.7877	37.4875	37.0543
1	47.3176	44.2186	43.8854	43.1190
2	47.0510	45.9513	45.5182	44.5518
3	46.2179	44.9517	45.4515	44.6518
4	47.4842	45.8514	45.6848	43.9520
5	47.5841	46.3512	45.0183	44.8517
6	46.7511	45.6514	45.0183	44.6185
7	46.6178	45.4182	43.2856	42.7857

Table 9.11 Test error using PCA+KNN and Test error using KNN with original data (Artificial).

What can be seen from table 9.11 is that the slightest error using PCA + KNN is a 42.7857 % while KNN using original data catch a 36.7877 %.

Neural network and PCA error comparison.

Then it will be represented, with the help of a table of results, the difference between performing dimensionality reduction using a neural network or using the PCA method.

Elements \ Neighbours	1	3	5	7
1	1,9994	-0,6498	7,0977	-0,8663
2	7,8307	7,9973	8,3306	7,9973
3	9,0636	8,064	9,93	9,6968
4	8,2306	8,6638	9,8967	8,7304
5	13,7287	10,4931	9,3635	9,7634
6	41,8194	41,2195	40,5198	40,3533
7	44,5185	43,4855	41,2196	40,7197

Table 9.12 Error difference committed by neural network and PCA (Artificial).

In each square the number represents the difference between the errors KNN collected by using data obtained from the reduction of attributes using the PCA method or neural network. The higher the number is, the more effective the neural network attribute reduction will be, when comparing to PCA.

9.2.2.4 Results of the Estate database.

In this section are the results obtained for the study of dimensionality reduction using the Estate data set.

Neural network error.

It contains the results obtained by applying a neural network in which we have been varying the number of the hidden layer neurons.

Neurons	Test Error %
1	11.3409
2	11.1529
3	11.4035
4	11.3409
5	11.1529
6	11.1529
7	11.3409

Table 9.13 Test error by using a neural network (Estate).

It is noted that the lower percentage of error obtained for the set of test data is an 11.1529 %, captured when the number of hidden neurons are 1, 5 or 6.

Error occurred when reducing dimensionality using a neural network.

The results are expressed as a percentage of error that makes a KNN classifier, to which rather than be provided with the original database, it is provided with the transformed data which has been reduced using a neural network.

Neurons \ Neighbours	1	3	5	7
1	19,9875	14,3484	12,5313	11,7794
2	19,7368	13,7845	11,9674	12
3	20,1754	14,2857	12,8446	11,4821
4	18,985	13,7845	12,2807	11,7794
5	19,4862	12,9699	12,4687	11,7794
6	18,7343	13,4085	12,218	11,6541
7	18,609	13,4085	12,4387	11,7794

Table 9.14 Test error using a neural network + KNN (Estate).

At it is shown in table 9.14 the lowest percentage of error made is 11.4821 %, collected in the test that reduces to 3 elements using a neural network, and using 7 neighbours in KNN classifier.

Error occurred when reducing dimensionality using PCA.

In this section are the results, expressed as a percentage of error, which makes a KNN classifier when using the N best elements of the database decided by PCA method.

In order to make a better comparison with the results obtained, the section contains the results obtained by applying KNN directly to the original data.

Components \ Neighbours	1	3	5	7
Original data	18.5464	13.5338	12.0927	11.4035
1	17,7825	14,3484	12,594	11,6541
2	18,1704	14,2857	12,2807	11,9048
3	17,8571	14,0351	12,1554	11,7794
4	35,6516	13,8471	12,3434	11,7168
5	17,9198	13,4085	12,1554	11,7168
6	18,4837	13,5965	12,218	11,5288
7	17,8571	13,2832	12,2807	11,7168

Table 9.15 Test error using PCA+KNN and Test error using KNN with original data (Estate).

What can be seen from table 9.15 is that the slightest error using PCA + KNN is an 11.5288 % while KNN using original data catch an 11.4035 %.

Neural network and PCA error comparison.

Then it will be represented, with the help of a table of results, the difference between performing dimensionality reduction using a neural network or using the PCA method.

Elements \ Neighbours	1	3	5	7
1	-2,205	0	0,0627	-0,1253
2	-1,5664	0,5012	0,3133	-0,0952
3	-2,3183	-0,2506	-0,6892	0,2973
4	16,6666	0,0626	0,0627	-0,0626
5	-1,5664	0,4386	-0,3133	-0,0626
6	-0,2506	0,188	0	-0,1253
7	-0,7519	-0,1253	-0,158	-0,0626

Table 9.16 Error difference committed by neural network and PCA (Estate).

In each square the number represents the difference between the errors KNN collected by using data obtained from the reduction of attributes using the PCA method or neural network. The higher the number is, the more effective the neural network attribute reduction will be, when comparing to PCA.

9.2.2.5 Results of the Splice database.

In this section are the results obtained for the study of dimensionality reduction using the Splice data set.

Neural network error.

It contains the results obtained by applying a neural network in which we have been varying the number of the hidden layer neurons.

Neurons	Test Error %
5	29.3333
10	25
15	29.3333
20	32
25	28
30	31
35	36.3333

Table 9.17 Test error by using a neural network (Splice).

It is noted that the lower percentage of error obtained for the set of test data is a 25 %, captured when the number of hidden neurons are 10.

Error occurred when reducing dimensionality using a neural network.

The results are expressed as a percentage of error that makes a KNN classifier, to which rather than be provided with the original database, it is provided with the transformed data which has been reduced using a neural network.

Neurons \ Neighbours	1	3	5	7
5	29,667	29,3333	26	28,3333
10	25,3333	25	26	25,6667
15	32,667	30	29	29,3333
20	29,3333	31,3333	33	32
25	33,6667	32,6667	31	31
30	38,6667	42,3333	39,6667	37,3333
35	35,3333	33,6667	32,6667	30,667

Table 9.18 Test error using a neural network + KNN (Splice).

At shown in table 9.18 the lowest percentage of error made is 25 %, collected in the test that reduces to 10 elements using a neural network, and using 3 neighbours in KNN classifier.

Error occurred when reducing dimensionality using PCA.

In this section are the results, expressed as a percentage of error, which makes a KNN classifier when use the N best elements of the database decided by PCA method.

In order to make a better comparison with the results obtained, the section contains the results obtained by applying KNN directly to the original data.

Components \ Neighbours	1	3	5	7
Original data	32	32	33.3333	33.6667
5	50,3333	47	45	46,3333
10	43	43,6667	44,3333	45,6667
15	45,3333	45,6667	43,3333	41,6667
20	42	41,6667	42,6667	41
25	42,6667	39	41	39,6667
30	35,3333	36,3333	35	35
35	29,6667	30,3333	27,6667	27,3333

Table 9.19 Test error using PCA+KNN and Test error using KNN with original data (Splice).

What can be seen from table 9.19 is that the slightest error using PCA + KNN is a 27.333 % while KNN using original data catch a 32 %.

Neural network and PCA error comparison.

Then it will be represented, with the help of a table of results, the difference between performing dimensionality reduction using a neural network or using the PCA method.

Neighbours / Elements	1	3	5	7
5	20,6663	17,6667	19	18
10	17,6667	18,6667	18,3333	20
15	12,6663	15,6667	14,3333	12,3334
20	12,6667	10,3334	9,6667	9
25	9	6,3333	10	8,6667
30	-3,3334	-6	-4,6667	-2,3333
35	-5,6666	-3,3334	-5	-3,3337

Table 9.20 Error difference committed by neural network and PCA (Splice).

In each square the number represents the difference between the errors KNN collected by using data obtained from the reduction of attributes using the PCA method or neural network. The higher the number is, the more effective the neural network attribute reduction will be, when comparing to PCA.

9.3 Conclusions.

In this section are de conclusions of the research. For this target, we decided to put all the best results in two tables to see if the system can work in an appropriate way.

At the next table we can see the best errors collected.

	KNN (Original data)	PCA+KNN (Components)	NN+KNN (Elements)	Hidden layer improves PCA
Artificial Domain	36,8% (10)	42,8% (7)	1,9% (7)	26 of 28
Spambase	10,6% (57)	11,9% (26)	7% (6)	28 of 28
German	29,3% (20)	29,3% (8)	24,3% (14)	13 of 28
Estate	11,4% (12)	11,5% (6)	11,5% (3)	11 of 28
Splice	32% (60)	27,3% (35)	25% (10)	20 of 28

Table 9.21 Best errors collected with NN and KNN classification.

Results table can conclude than performing the dimensionality reduction using a neural network is, for most cases, a very valid option, because lower error rates are reported in most test, and those in which it is not, the error is very close and therefore may be worth realize the reduction, if we can afford that little mistake added.

According to the results collected in the dimensionality reduction using PCA, does not perform a very good results when a KNN classify the new data, and in no way improves the neural network results, so between the two methods, dimensionality reduction winner is the neural network dimensionality reduction.